42-Volt PowerNet System Management Using Multiplexed Remote Switching

by

James Russell Geraci

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Engineering

and

Bachelor of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1999 [June 1999]

Author ______________________________
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by ______________________________
Dr. Tom Keim
Research Scientist
Thesis Supervisor

Certified by ______________________________
John G. Kassakian
Professor
Thesis Supervisor

Accepted by ______________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Students

42-Volt PowerNet System Management Using Multiplexed Remote Switching

by

James Russell Geraci

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 1999, in partial fulfillment of the
requirements for the degrees of
Master of Engineering

and

Bachelor of Science in Electrical Engineering

Abstract

The main objective of this thesis is to explore techniques for using multiplexed remote switching in a 42/14 volt dual voltage automotive environment to perform bus energy management and other useful system functions. Achieving this objective involved first constructing a 42v/14v dual voltage automotive test facility. Then, designing and evaluating candidate algorithms for bus energy management in a dual-voltage electrical system using that test facility. The energy management algorithms explored in this thesis were designed to minimize the cost and equipment needed to implement the algorithms. This will allow future work to perform cost vs. performance gain analysis.

Thesis Supervisor: Dr. Tom Keim
Title: Research Scientist

Thesis Supervisor: John G. Kassakian
Title: Professor

Acknowledgments

I would like to thank Professor John Kassakian and Dr. Tom Jahns for originally entrusting me with this project. I would also like to thank Dr. Tom Keim for his insight and help with this project. I would like to thank my parents and brothers and sisters for their support. I would also like to thank Li Yu for her support and help with some of the most physically demanding parts of the project. Finally, I would like to thank Ed Lovelace and Dan Santos for their work in obtaining a drive motor for my project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0.1 Project Overview

The objective of this thesis project was to explore techniques for using multiplexed remote switching in a dual-voltage system to perform bus energy management and other useful system functions. "Multiplexed remote switching" is a term used to describe the ability of an in-car computer network to control the state of various loads within the automobile. Such a system would require a data network, several microcontrollers, and switches whose state can be controlled by the microcontrollers. Because of the ever increasing amout of wiring in automobiles, the next generation automobile electrical system will have such a remote switching network installed. Figure 1.1 shows the main parts one possible topology for the next generation automotive electrical system. It is a 42/14 volt unidirectional DC/DC converter based automotive electrical system.

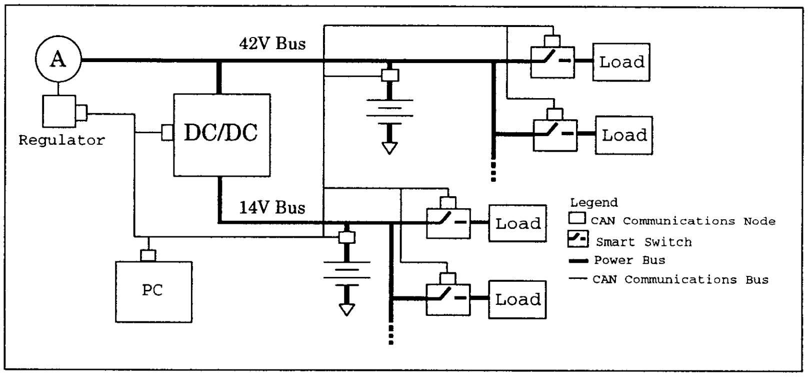


Figure 1.1: Dual Voltage Architecture with Communications Bus

In this dual voltage environment there are two voltage busses, a 42 volt and a 14 volt bus. Loads are attached to each bus and their on/off state is controlled by a microcontroller controlled switch. An example of a 42 volt load would be a front windshield heater. A 14 volt bus load might be the dome light that turns on when the car doors are opened. A complete list of loads used for this thesis can be found in Table 3.2.1.

There are three sources of electrical power in the system of Figure 1.1. The first is the alternator and the others are the batteries. When the gasoline engine is running, it turns the alternator which converts the mechanical power of the engine into the electrical power used to supply the electrical system of the car. The batteries have different functions depending on if the car is running (key-on) or not(key-off). When the car is running, the two batteries perform a load leveling function. They provide power to their respective busses when the total demand for power on a bus exceeds the amount that is being provided to that bus by the alternator. When the car is off, each battery has a different function. The 42 volt battery's function is to start the car. The 14 volt battery's function is to ensure that the key-off loads have power during the entire time the car is off. The DC/DC converter acts as a regulated valve controlling power flow between the two busses.

If size, weight, and money were not an issue, the alternator should be sized so that it would be able to provide enough power so that there would be no possible combination of loads which could drain the batteries. Because of physical and economic limitations, however, such an alternator is not obtainable. Furthermore, such an alternator might not be the most desirable alternative. Due to the start and stop nature of automobile driving, there are times when the car batteries are being drained and times when they are being charged. The important thing is that the change in state of charge of each battery over the complete drive cycle is zero or positive. If it were possible to intelligently control the flow of charge between the two batteries so that no net charge is lost by either battery over a given drive cycle, it would be possible to size the alternator so that it would not have to provide enough power to keep both batteries fully charged at all times. This method of intelligently controlling the flow of energy throughout the automobile is called active energy management. Such an energy management system would allow the use of a smaller alternator and therefore reduce the weight and cost of the automobile.

It is highly likely that the next generation of automobile electrical system will include a multiplexed remote switching network. If it does include such a network, then the system will have the necessary communications and control elements to perform not only the communications necessary for an energy management algorithm to work but also to perform the computations necessary to make intelligent decisions based on the state of the automobile's electrical system. It is the purpose of this research to use a multiplexed remote switching network to investigate the performance of a number of energy management algorithms.

Chapter 2

Energy Management Algorithms

Energy management involves the estimation of energy consumption, proper sizing of equipment to meet this estimate, and proper operation of the equipment [1]. Energy management algorithms are a way to control the flow of energy throughout an automobile's electrical system. All energy management algorithms take in information about the system's state in order to try to determine the state of charge of the batteries. State of charge is a term often used to refer to the amount of work that the battery can do given an instantaneous set of environmental parameters[1]. In addition, each algorithm can be customized to not only take into account information about the state of the system but also take into account safety information and preferences which might be of most benefit to the vehicle operator. For example, in the case of the energy management algorithms developed for this thesis, a strong preference was given to the operator being able to start his car. The system then combines the physical information and the preference information and uses that information to appropriately modify the state of the system's energy sources and sinks.

2.1 Present Energy Management System

Energy management algorithms are not new to the automobile industry. Today's automobile employs a simple yet effective energy management algorithm. It uses a voltage sensor that has a temperature compensated output voltage to measure the battery's voltage and uses this information to control the excitation of the alternator field winding, and thus the amount of power that the alternator will deliver to the system. This energy management algorithm uses curve A from Figure 2.1 as it's battery model [2].

Curve A in Figure 2.1 is a graph of battery cell voltage versus time for a battery which is slowly being drained at a constant current. Because batteries are rated in amp·hours, if the total charge leaving a battery is measured and the initial state of the battery is known, the state of charge

[1] Not all algorithms actually calculate a state of charge. Most take action based on physical indicators necessary to compute the state of charge, but do not actually compute the state of charge itself

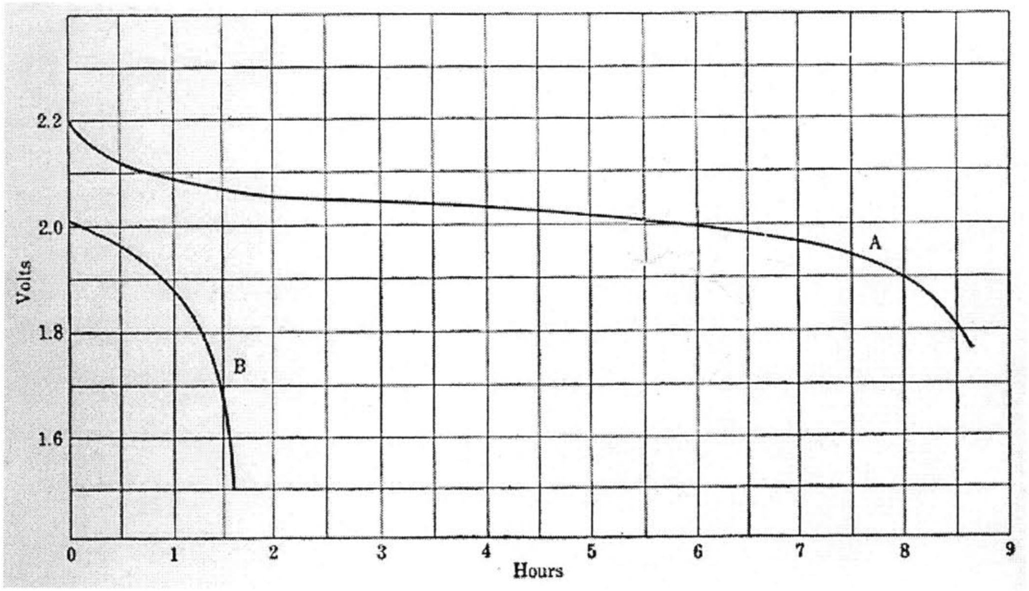


Figure 2.1: Typical Voltage-time Discharge Curves of Lead Acid Cells

of the battery can be computed. By using this graph, a relationship between the voltage of the battery and the battery's state of charge can be made. The present system of observing the bus voltage and then modifying the alternator excitation accordingly is simply trying to use the voltage information to make a guess at how much charge has been removed from the battery during a drive cycle. This algorithm does not compute a number for the state of charge, but simply reacts to the voltage which is an indicator of the state of charge of the battery.

2.2 42V/14V Energy Management System

The 42V/14V electrical system will also employ an energy management algorithm; however, the fact that there are now two batteries makes the control of the system more complex and the possible benefits of having a good energy management algorithm greater. This thesis three main levels of sophistication for an energy management algorithm.

1. Bus Voltage Regulation

2. Sophisticated Battery Model

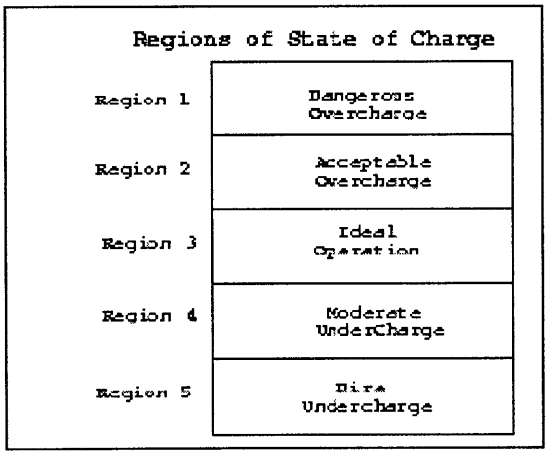
3. Artificial Intelligence

2.2.1 Bus Voltage Regulation

Bus voltage regulation is the 42V/14V extension of the present day energy management algorithm. It employs a temperature compensated voltage sensor on the outputs of the DC/DC converter and the 42V alternator to measure the voltage on each bus and then uses curve A in Figure 2.1 to infer the state of charge of each battery. It has the advantage that it can be easily implemented and can be expected to maintain battery charge for both batteries about as well as today's highly satisfactory system.

2.2.2 Sophisticated Battery Model

The second level of sophistication employs a more sophisticated battery model than the bus voltage regulation level. This level employs state of charge explicitly rather than implicitly through bus voltage. By reasoning about battery state of charge directly, it becomes possible to make more intelligent decisions about how to control the states of the energy sources and sinks on the network and thus develop an energy management algorithm. One way to use state of charge information to help develop an energy management algorithm is to first break each battery's state of charge into a number of different regions and then make decisions based on which region each battery is in at any given time. An example of how a battery's state of charge might be decomposed into different regions is shown in Figure 2.2.



Regions of State of Charge

Region 1	Dangerous Overcharge
Region 2	Acceptable Overcharge
Region 3	Ideal Operation
Region 4	Moderate UnderCharge
Region 5	Dire Undercharge

Figure 2.2: Battery State of Charge Partitioning used for this Thesis

Figure 2.2 shows the battery state of charge broken into 5 different regions. The exact place in the state of charge continuum where each of the regions starts and stops have not yet been standardized; however, for the purpose of this thesis, the following divisions were created:

- Region 1: Dangerous Overcharge → $115\% \leq \text{SOC}$
- Region 2: Acceptable Overcharge → $105\% \leq \text{SOC} < 115\%$
- Region 3: Ideal Operation → $90\% \leq \text{SOC} < 105\%$
- Region 4: Moderate Undercharge → $50\% \leq \text{SOC} < 90\%$
- Region 5: Dire Undercharge → $\text{SOC} < 50\%$

Figure 2.3 shows the 5x5 decision matrix which graphically displays the 25 different possible regions into which the states of charge of both batteries may fall. The numbers on each edge correspond to the state of charge regions in Figure 2.3.

A few examples of possible decisions based solely on the state of charge of the batteries are written in the boxes in Figure 2.3. If both batteries are in a dangerous state of overcharge, then the algorithm would turn off the DC/DC converter, decrease the alternator field winding excitation (possibly turning it off), and turn on select high power loads on both the 42v bus and the 14v bus. These actions would immediately cut off power flow into the 12v battery, so it would begin to discharge. It would also allow the 36v battery to begin discharging as rapidly as possible. This kind of situation would not occur in the voltage regulation energy management system unless something had gone wrong with the voltage regulators, so actions taken during this mode of operation can be seen as a sort of a safety device.

Another situation the system might get in is if both batteries are in a dire state of undercharge. This situation might occur if, over a period of time, both batteries are drained and not returned to a full state of charge after each drive cycle. In such a situation there might be the possibility of recharging one of the batteries. This is where the engineer must make a decision as to what action would best serve the customer. The system could either be designed to let the DC/DC converter try to regulate the 14v bus and thus hopefully save the 12v battery, or it could be designed to shut the DC/DC converter off and hopefully save the 36v battery.

36V Battery SOC

12 Battery SOC	1	2	3	4	5
1	DC/DC OFF Alternator Down		DC/DC OFF		DC/DC OFF
2					
3	Alternator Down		No Action		DC/DC Down
4					
5	DC/DC Full ON		DC/DC Full ON		Alternator Full On DC/DC OFF

Figure 2.3: Decision chart based on state of charge

If, in addition to the state of charge information, the current on each battery were known, then even more informed decisions could be made. For example, if both batteries were in a region of acceptable undercharge, but the 12v battery was draining, while the 36v battery was being charged, the system could be designed so that the DC/DC converter would pass more current to the 12v battery without causing the 36v battery to drain. This would keep the 36v battery in an acceptable region of charge and it would either minimize the rate at which the 12v battery discharged, thus extending the life of the 12v battery, or it might allow the 12v battery to begin charging. It might even be possible for both batteries to charge. For example, if the 36v battery were charging at a rate of 6 amps, and the 12v battery were discharging at a rate of 5 amps, it might be possible to control the DC/DC converter so that the 36v battery would charge at a rate of 3 amps and the 12v battery would charge at a rate of 4 amps.

The benefit of the sophisticated battery model energy management algorithm, over the simple voltage regulation algorithm, is that the designer of the electrical system has more flexibility to dictate how the system responds to different loading states. Because this algorithm can limit the amount of current delivered by the DC/DC converter, it is possible to charge the 12v battery at a rate that is less than the converter's maximum current delivery capability. With reduced output, the current drawn from the 42V bus by the converter is reduced. This current can instead go to the 36V battery thus reducing its rate of discharge and possibly even allowing it to charge. Therefore, the situation could exist where both batteries are charging, albiet very slowly, instead of in the voltage regulation case where only one battery is charging rapidly and the other is draining because it is feeding the charging battery.

2.2.3 Artificial Intelligence

The decisions made by the energy management algorithm become the most helpful when the system is aware of the physical environment around the car and can possibly learn the operator's driving habits. Such a system might be aware of the date, the time of day, and the outside temperature. It could be made aware of the weather forecast by having it automatically dial into the weather service each night so it could adjust how it behaves for the following day. It could also be plugged into a GPS system. If it then knew its starting point and its finish it could calculate the amount of time that it would be driving and possibly the type of driving (in city or country) that it would be doing. This information could have a significant impact on the way that energy is managed in the system. Take again, for example, the situation where both batteries are in an acceptable state of undercharge and the 12v battery is discharging and the 42v battery is charging. If, the car knew that it was going to be doing a short drive and that the 12v battery wasn't discharging too rapidly, it might choose to decrease the output of the DC/DC converter so that the 12v battery drained a little more rapidly, but the 36v battery would charge more rapidly and might possibly move into a region of ideal operation.

Finally, the decision on how to control the DC/DC converter would change once again if the car were able to learn the driver's driving habits. If, for example, it were Friday at 6PM and the car knew that the driver always went to his cabin for the weekend, and that the driver just let his car sit over the weekend, the car would want to try to maximize the charge on the 12v battery by increasing the output of the DC/DC converter so that it would be able to power all of its key-off loads for the weekend.

2.2.4 Tested Energy Management Algorithms

For the purpose of this thesis both the 42v/14v bus regulation algorithm and a sophisticated battery model algorithm were tested. Information about the load cycles, drive cycles, and physical test facilities used to test these energy management algorithms can be found in Chapter 4. The sophisticated battery model algorithm was limited to controlling only the state of the DC/DC converter.

2.2.4.1 42v/14v Bus Regulation Algorithm

The 42v/14v bus regulation algorithm which was tested simply used the voltage regulators on the DC/DC converter and the alternator to control the flow of power throughout the system. Figure 2.4 shows the regulation characteristic of the DC/DC converter. This curve means that the DC/DC converter will try to deliver it's maximum current of 68 amps anytime the voltage on the 12v battery drops below 13.8 volts. Figure 2.5 shows the alternator's regulation characteristic. The alternator is set to regulate its output to 40 volts, and it can deliver up to 90 amps in order to maintain a 40 volt output voltage.

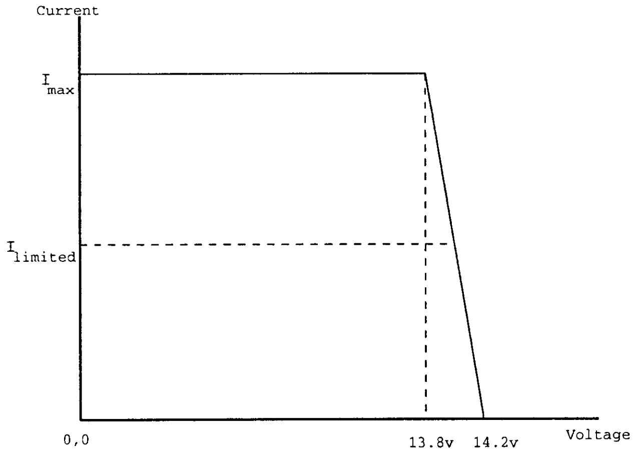


Figure 2.4: Regulation Curve for DC/DC converter

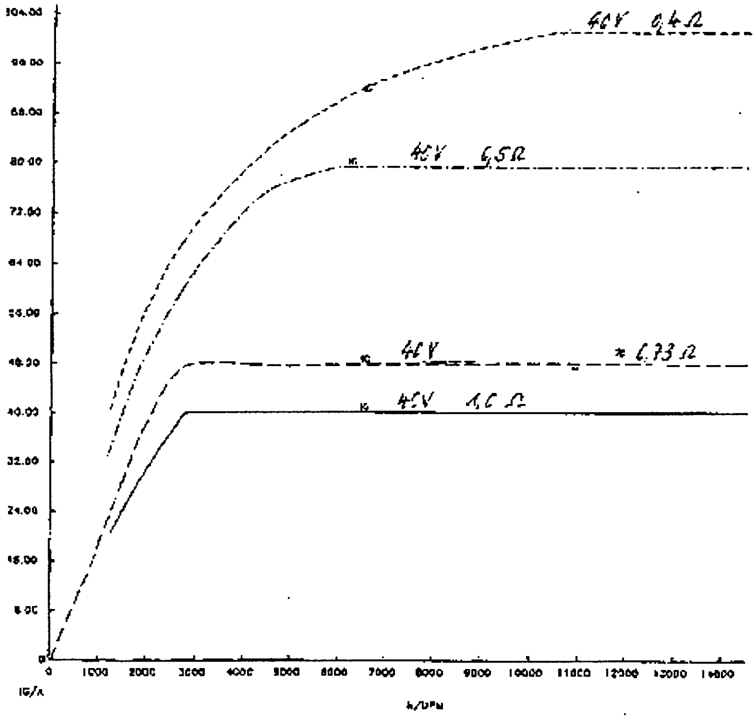


Figure 2.5: 40v alternator Current vs. RPM Characteristic

2.2.4.2 Sophisticated Battery Model Algorithm

The sophisticated battery model algorithm which was designed was based on a coulomb counting algorithm. The amount of current coming out of each of the batteries was measured about once every second and its integral was computed. This value was then used to compute the percent state of charge according to Formula 2.1.

$$\text{State of Charge} = \frac{(\text{Initial Amp}\cdot\text{hours}) - (\text{Amp}\cdot\text{hours used})}{(\text{Initial Amp}\cdot\text{hours})} \qquad (2.1)$$

Once the state of charge for each battery was calculated[2], the system's present operating region in Figure 2.3 was determined. From there, current information was used to make a final decision about the state of the DC/DC converter. A complete enumeration of all possible decisions can be found in Appendix A

Long-term inaccuracies in the discrete approximation of the total change in charge of a battery will result in the true state of charge diverging over time from the state of charge calculated the present test facilities data collection equipment. Even if it were possible to count every coulomb entering and leaving the battery, the calculated state of charge and the true state of charge would diverge due to internal self-discharge mechanisms. Over a long period of time, any control algorithm that computes battery state of charge soley on equation 2.1 would need to be supplemented by additional information sensitive to actual state of charge, for example, voltage and temperature. For the purpose of this thesis, however, the rate of divergence between calculated and actual state of charge should be slow enough to permit meaningful observations.

[2] State of charge will be used to mean percent state of charge from now on.

Chapter 3

MIT Breadboard Facility

In order to validate the energy management algorithms that were discussed in Chapter 2, it was necessary to construct physical test facilities on which those tests should be conducted. The facility that was to be constructed had to be an easily controllable and modifiable electrical equivalant of the 42V/14V unidirectional DC/DC converter architecture from Figure 1.1. the facility can be broken down 3 major parts.

1. Power Delivery Systems: Section 3.1
   - The Breadboard Power Cabling: Section 3.1.1
   - The Breadboard Batteries: Section 3.1.2
   - The Breadboard Alternator and Support Hardware: Section 3.1.3
   - The Breadboard DC/DC Converter: Section 3.1.4
2. Power Dissipating Systems: Section 3.2
   - Fixed Resistance Loads: Section 3.2.1
   - Speed Dependent Loads: Section 3.2.2
3. Control Systems: Section 3.3
   - PC Master Control System: Section 3.3.1
   - The C167CR: Section 3.3.2
     - The CAN Bus: Section 3.3.2.1
   - Data Collection System: Section 3.3.6
   - Software to generate PC input files: Section 3.3.7

3.1 Power Delivery Systems

The breadboard power delivery system is made up of all sources of power and the physical cabling necessary to deliver that power to the systems loads. This includes the batteries, the alternator and its support equipment, the DC/DC converter and the cables necessary to deliver power to the loads.

3.1.1 The Breadboard Power Cabling

A diagram of the power cabling for the MIT breadboard facility can be seen in Figure 3.1.

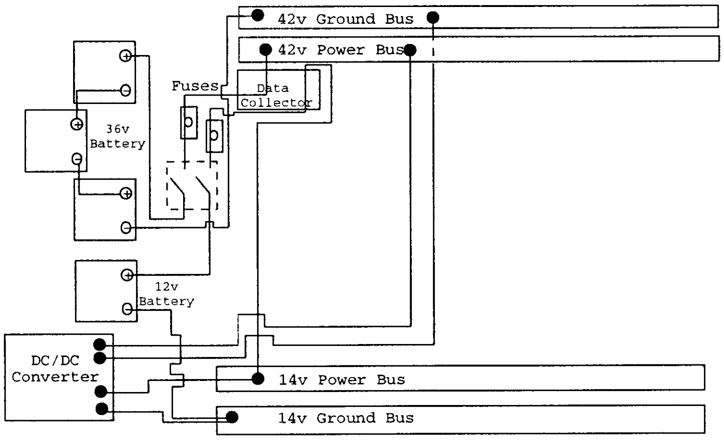


Figure 3.1: Diagram of MIT Breadboard Facility

Each power and ground bus was implemented by an aluminum rail. The two power busses are located on opposite sides of the breadboard facility. Leads from loads can be screwed to each of the rails. There are two separate ground rails. These represent different local grounds that might occur in an automobile. They are connected togther by a pair of 4 AWG cable. This pair of cable performs the same function as that of a chasis in an automobile.

3.1.2 Breadboard Batteries

The 36V battery was made up of 3 AC Delco Professional Freedom Car and Truck 58-5YR batteries connected in series. They have a reserve capacity[1] of 70 minutes. The 12V battery was

[1] Reserve Capacity [3] is the ability of the battery to maintain a cell voltage of 1.75V or greater at a discharge rate of 25 amps.

an AC Delco Professional Freedom Car and Truck 65-7YR battery. It has a reserve capacity of 160 minutes.

3.1.3 The Breadboard Alternator

The alternator used to provide power to the network was a 40V Bosch alternator that was given to the MIT Constorium for Advanced Automotive Electrical And Electronic Equipment by Paul Nicastri of Ford. The alternator can supply 50 amps at idle and 90 amps at higher rpm. Thus the alternator can supply a maximum of 2000 watts at idle and 3600 watts at higher rpm. Its output current vs. rpm characteristic can be seen in Figure 2.5. The appropriate wiring diagram for the alternator can be seen in Figure 3.2.

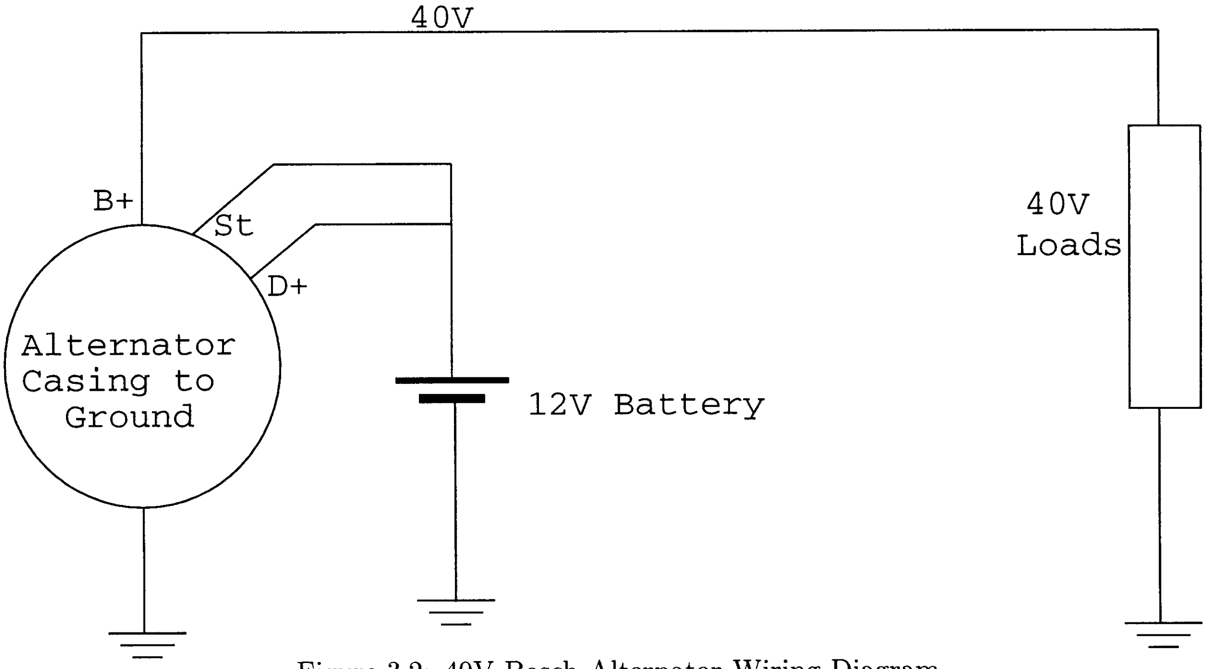


Figure 3.2: 40V Bosch Alternator Wiring Diagram

In a conventional automobile, the alternator is spun by the car's engine. It is geared at a ratio of approximately 3 alternator rotations for every one engine rotation. The situation is the same with the breadboard facility. The alternator was controlled by an 18hp 13.4kW Pacific Scientific PacTorq Brushless P.M. Servomotor. The servomotor and the alternator were geared so that one rotation of the servomotor produces about 3 rotations of the alternator. The speed of the motor was controlled by a Pacific Scientific 756 ServoController. The appropriate wiring of the 756 ServoController to the PacTorq servomotor can be seen in Table 3.1. The controller is controlled through its serial port, and for testing purposes , it is being software limited by its control program, 'PacTorq.bas'[2], to spinning the PacTorq motor to 3500rpm. If this limit is exceeded, the motor stops all motion and cannot move again until it is reprogrammed.

Power Connections	
PacTorq Motor Connection Label	SC756 Drive Connection Label
T_1	T
T_2	R
T_3	S
Resolver Connections	
Pactorq Motor Connection Number	SC756 Drive J51 Connection Number
1	4
2	3
3	2
4	1
NONE	5
5	6
6	7
7	NONE
8	8
9	9
10	NONE

Table 3.1: PacTorq Motor to SC756 Motor Driver Wiring Connections

3.1.4 The Breadboard DC/DC Converter

The breadboard's DC/DC converter is a unidirectional converter that is capable of delivering up to 68 amps to the 14v bus. It's regulation characteristic can be seen in Figure 2.4. The DC/DC converter can be controlled to deliver an amount of current less than its instantaneous maximum deliverable power. An example of this can be seen in Figure 2.4. In Figure 2.4 the converter can supply I_{max} but it can also supply any amount of current less than I_{max} like $I_{limited}$ for example. The converter, however, cannot be controlled to deliver an amount of current greater than its regulation characteristic will allow. For example, if the 14V bus were at 14.0V (it is regulated to 14.2v) then the maximum amount of current that the converter could deliver is 34 amps. It cannot

[2] 'PacTorq.bas' can be found in Appendix B

be controlled in any way to deliver more than 34 amps, but it can be controlled to deliver any amount of current less than 34 amps.

The current limit of the DC/DC converter can be set by changing the value that appears on its 8-bit input seen in Figure 3.3.

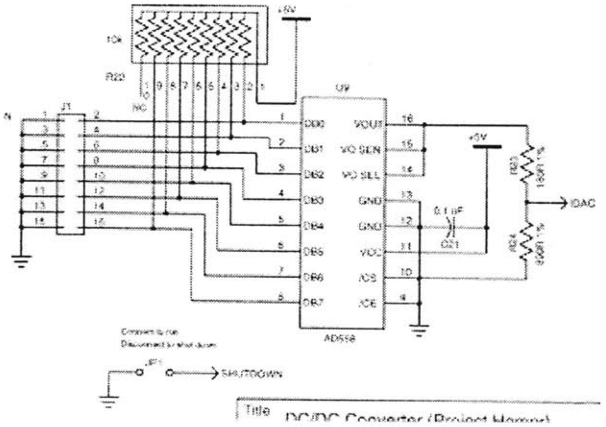


Figure 3.3: Digital Input of the MIT Breadboard DC/DC Converter

Each input pin of the AD558 A/D converter has a pull up resistor. The pin can be brought to logical low by first connecting an open drain configured transistor to the resistor and then activating that transistor. The converter is at maximum current when all of the pins are high, and it is at zero current when all the pins are low. Pin DB0 on the AD558 is the LSB. The on/off state of the converter is controlled by a separate pin. The converter will turn on when this pin is connected to ground.

3.2 Power Dissipating Systems

By the year 2005 some automobiles will have an average electrical load of over 2500 watts [4]. The electrical loads for the breadboard were selected in order to allow loading in excess of 2500 watts. The loads that were selected for the breadboard facility can dissipate a total of about 4100 watts. This is well above the maximum alternator output of 3600 watts at high alternator rpm. Therefore, because the batteries must be used, an energy management algorithm is relevant.

In the case of the breadboard facility, loads can be broken down into two different categories. The first type of load is a fixed resistance load, and the second type of load is a speed-dependent load. For the MIT breadboard facility 11 different fixed resistance loads were selected and implemented as CAN enabled smart switch controlled loads. The electromechanical valve system was the only speed-dependent load enabled on the breadboard. It is discussed in Section 3.2.2.

3.2.1 Fixed Resistance Loads

The loads that were selected as fixed resistance loads are shown in Table 3.2.1. The resistors were held in aluminum mounts and power flow to the resistors was controlled by a microcontroller controller power MOSFET. The Siemen's BTS550P was used to switch on and off loads on the 14V bus, and the Siemen's BTS660P was used to control loads on the 40V bus. Each MOSFET provides as an output on one of its pins a current that is proportional to the amount of current flowing through its channel. The MOSFETs were mounted to custom designed boards. Also mounted to each board was a LM317 voltage regulator that was used to provide power to the CAN microcontroller that was controlling the state of the MOSFET via instructions it was receiving over the CAN bus[3] A circuit diagram for the BTS660P's board can be seen in Figure 3.4, and a circuit diagram for the BTS550P's board can be seen in Figure 3.5.

3.2.2 Speed Dependent Loads

The electromechanical valve system was the only speed-dependent load enabled on the breadboard. It was implemented using a Hewlett Packard 6050A 1800Watt Programmable Load that was configured to draw a current proportional to the speed of the alternator. The amount of current it demanded was varied with alternator speed according to Equation 3.1. It has a minimum demand of 9 amps at idle (alternator speed of 1800 rpm) and a maximum of 45 amps at higher speeds (alternator speed of 6000 rpm or more). The HP 6050A received control commands over a GPIB bus.

[3] See Section 3.3.2 for a detailed description of the CAN bus.

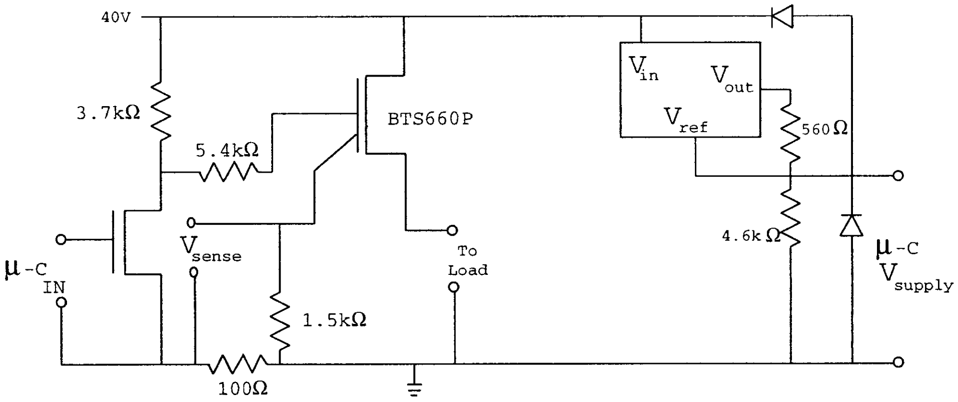


Figure 3.4: Circuit Diagram of BTS660P Smart Switch Board

$$\mathrm{I}_{demanded} = \frac{9}{350}\mathrm{Motor}_{rpm} - 6.425 \tag{3.1}$$

3.3 Control Systems

3.3.1 PC Master Control System

Because the breadboard facility cannot be driven, a method of simulating driving had to be created. This virtual driver was implemented using LabView 5.0. The virtual driver was coded in LabView's multithreaded 'G' graphical programming language and run on a 200 MHz Pentium PC running Windows95. Figure 3.15 shows the final PC interface for the facility. The virtual driver had to be able to turn on and off fixed resistance loads, control the amount of current drawn by the DC/DC converter, control the speed of the alternator, and collect information about the state of the system. A subprogram was written to control each of these functions, and these subprograms were combined toghter in the file "testcircuit2.vi." The major subprograms[4] are shown in Figure 3.16. The current drawn by the DC/DC converter is controlled by 'EMValve.vi.' The speed of the alternator is controlled by 'PACSCIBYTE.vi', Information going to and received from the CAN bus is controlled by 'SerialController.vi.' Information is sent through the CAN bus to the PC, so

[4] Programs and subprograms are called 'VIs' in LabView

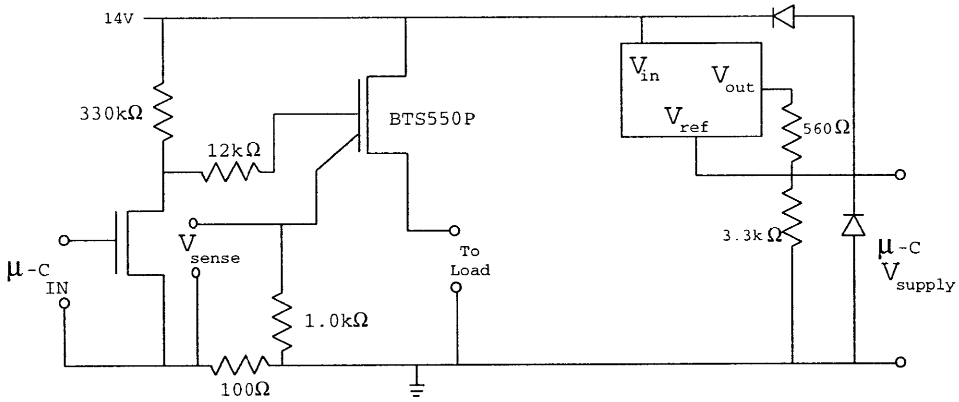


Figure 3.5: Circuit Diagram of BTS550P Smart Switch Board

the CAN bus is the means of collection of information about the state of the system.

3.3.1.1 LabView File Input

The virtual driver itself is implemented in 'fileinputtest2.vi'. Fileinputtest2.vi reads in a specially formatted file into a gian 2D array and then converts the information in the 2D array into information that in the appropriate 'vi' can use to create electrical events on the breadboard facility. This file is generated by a custom Java program that is described in Section 3.3.7. A few lines from one of these files can be seen in Figure 3.3.1.1.

```
!54 ?822 ^42+14.0 #A00300000500000000000000000000080A //
!55 ?1239 ^42+25.0 #A0030000090000000000000000000000C0A //
!56 ?1645 ^42+35.0 #A00300000500000000000000000000080A //
!57 #A0030000090000000000000000000000C0A //
```

Figure 3.6: A few lines from a breadboard input file

Fileinputtest2.vi parses each line of the breadboard input file into a number of different tokens. The information portion of each token is written to a global variable that has been designated as a holder of that token's information. This global variable is, in turn, read by the appropriate subvi. For example, take the line from Figure 3.3.1.1 that starts with "!54". This line would be broken into 4 different tokens. The first token starts with a '!'. This tells the file input subprogram that

Breadboard Loads				
14V Bus Loads				
Load Name	Saber Name	Wattage	Current	Resistance
Power Door Locks	sdr_locks	88	6.0	2.4
Seat and Door Module	sdr_seat_adjust	13	1	15
Turn Lights	sdr_turn	111	7.9	1.8
ABS	sdr_abs_tc	324	23	0.6
Brake Loads	sdr_brakes	146	10.5	1.3
42V Bus Loads				
Rear Seat Heater	sdr_rear_seat_htrs	180	4.29	9.78
Air Pump	sdr_emissions	480	11.4	3.7
Heated Windshield	sdr_windshield	700	16.7	2.5

Table 3.2: Fixed Resistance Breadboard Loads

the following information is the time offset, in seconds, since the start of the test. It is written to the global variable “Time Counter Global.” The next token, ’?’, tells fileinputtest2.vi that this information is the new speed, in rpm, of the PacSci Servomotor. Information following a ’?’ is written to global variable “RUNSPEED.” The alternator rotates at 3 times the value in this global variable. The third token “4̂2+” tells fileinputtest2.vi that this information is the new amount of current to be demanded by the programmable load. It amount of current to be demanded is written to global variable “E&M valve current demand.” In this case the amount of current to be demaded is 14 amps. The fourth token, “#” indicates that the following data is a CAN message. It is written to global variable “CAN write buffer.” The final token, “//” tells fileinputtest2.vi that this is the end of the line and that it should proceed to the start of the next line. It is important to note that not all lines will have all tokens, and, therefore, the length of the lines in the input file may vary. The line starting with “!57” only contains 3 tokens compared with the 5 of the line starting with “!54”. This helps greatly reduce the size of the breadboard input file and this in turn greatly improves the performance of the entire system because it allows better use of the host PC’s processing power. The LabView code that reads in the breadboard input file and parses it can be seen in Figure 3.7. The code consists of two large while loops and several inner condition statements. Every time through the inner loop consists of reading in and testing a token, i.e. reading in a single column element from a row and sending the information in the column element to the appropriate global variable. Execution of the outer loop corresponds to changing to a new row.

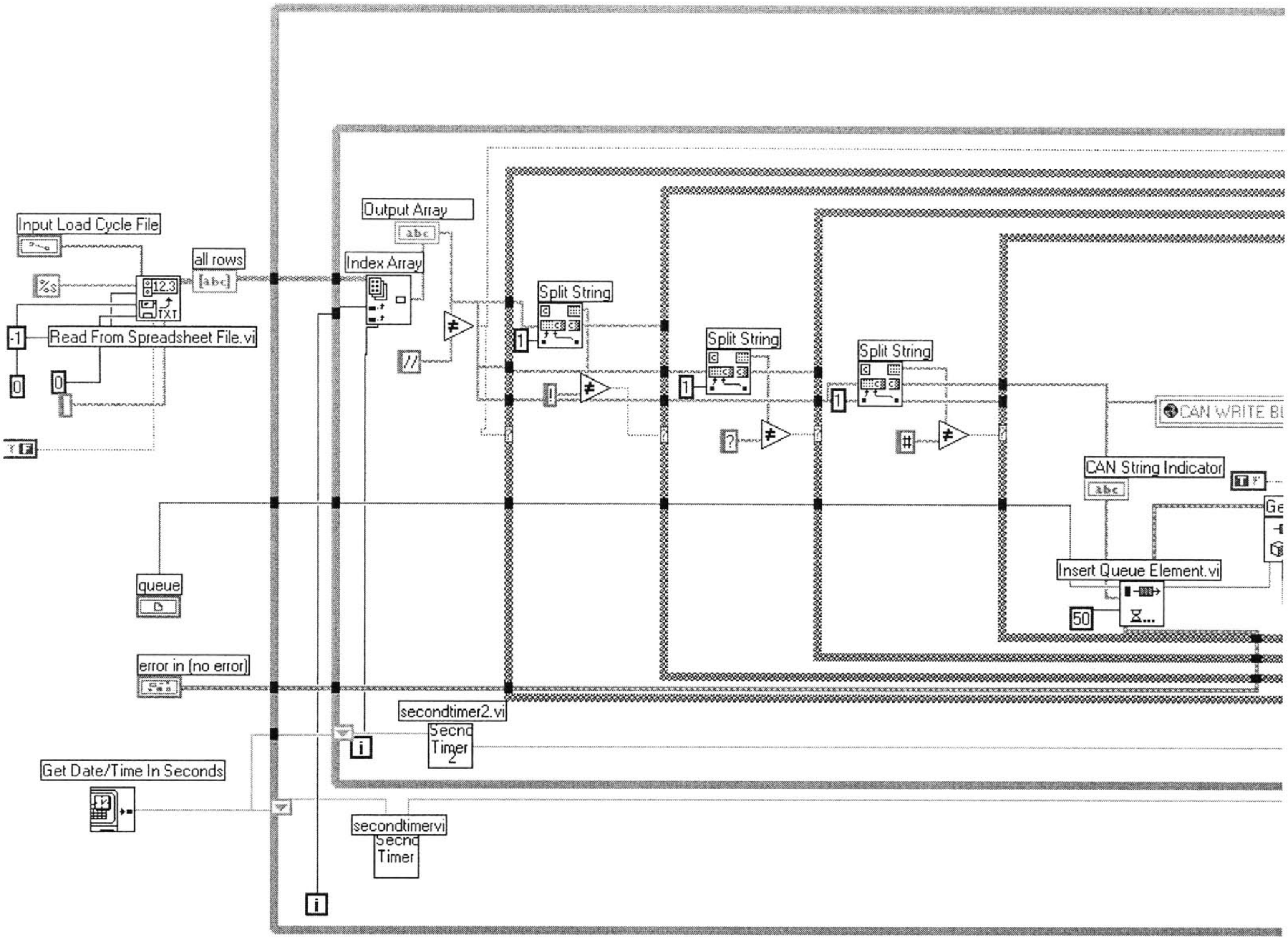


Figure 3.7: LabView 'G' code that parses breadboard input files

3.3.1.2 CAN Bus I/O

Unfortunately, there is no known CAN interface to LabView. In order to use a PC card that will allow the system to connect directly to the CAN bus, LabView would have to call a Windows dynamic linked library function. LabView is implemented so that when it calls a Windows dll, LabView stops all threads from executing until that dll function call is complete. This means that every time the system wants to watch activity on the CAN bus, or receive a piece of information from the CAN bus, LabView would have to stop all threads of execution and wait. If the CAN bus were accessed more than a few times a second, the system could quickly get bogged down. LabView does, however, have native serial port accessing methods, and it has serial port support though its native VISA[5] support. It was decided then, that the PC would be connected to the CAN bus through a serial router. Presently, this serial router only operates at 9600 baud; however, the serial router can be operated at baud rates up to 625KBaud. The operation of the router is described in Section 3.3.5.

3.3.1.3 Electromechanical Valve I/O

The electromechanical valve I/O subprogram was also implemented in VISA, and it's software is almost identical to that of the CAN I/O subprogram except for the fact that it only transmits data and never requests feedback from the programmable load. The programmable load has 3 different 600 Watt channels that can be controlled togther to give up to 1800 Watts. The electromechanical valve I/O subprogram divides the demand between the three channels evenly. Each channel never demands more than 15 amps individually.

3.3.1.4 Alternator Speed Control I/O

The alternator speed was controlled through communications port 1 (COM1) on the PC. It's interface program was written using LabView's VISA modules so it should run on Windows NT as well as Windows 95. It operates by sending a string through the serial port to the servocontroller that was controlling the speed of the alternator. For example, if it was desired to have the alternator spin at 900 rpm, then the string "00900" (plus a carriage return) was written to COM port 1. There are always 2 leading zeros because LabView uses one and the servocontroller uses the second one to create an interrupt to which it will respond. Therefore, the third value 9 is the first value read in by the servocontroller. The string "00900" will cause the servomotor to spin at 900 rpm. This means that the alternator is spinning at 2700 rpm.

3.3.1.5 User Interface Related Activities

The user interface is the lowest priority subprogram in LabView. Under heavy loading situations LabView will often not update the interface right away. This can give the appearance of a delay in the network; however, this is not the case. It is only LabView trying to make sure that all I/O subprograms operate properly even at the expense of the user interface. This portion of the

[5] VISA is an interface which allows you to access all of the PC's I/O ports in an identical fashion through generic Read/Write commands. Therefore, it is possible to use almost the same code to access a GPIB port as it is to access a serial port.

program is also responsible for writing collected data to the hard disk. The data that is written is battery voltage for each battery, time, and motor rpm.

3.3.1.6 LabView File Output

LabView takes bus voltage, alternator shaft speed and battery current information and writes it to an output file. By default, the file is named "output.txt" and it located in the root directory of the "d:" drive of the PC that was used. The output file is a tab delimited file. The columns in the file represent time, alternator shaft speed, 42V bus voltage, 14V bus voltage, 42V bus current, 14V bus current, state of DC/DC converter.

3.3.2 The CAN bus and the C167CR

One of the features of the next generation of automobile electrical system may very well be some type of multiplexed data network that will control the state of the loads. The breadboard facility implements this feature in the form of a CAN network. CAN is a Bosch networking protocol which was developed in the late 1980's for use in the automotive industry. CAN is an acronym which stands for Controller Area Network. A complete discussion of the specifics of the CAN network protocol can be found in the book "CAN System Engineering: From Theory to Practical Applications" [5]. CAN is a standard for transmitting messages, and the exact hardware implementation might vary between vendors. For the purpose of this thesis it is important to understand the Siemens C167CR microcontroller, and how Siemens implements the CAN protocol in this controller.

The C167CR microcontroller is a 16-bit microcontroller. The CPU is able to operate at clock speeds of up to 20 MHz. One of the major applications for microcontrollers is data collection and real time control of external systems. To better achieve this goal, there is an on chip peripheral subsystem that operates independent of the CPU core. This peripheral subsystem is connected with the CPU via a complex system of interrupts. If the peripheral needs the CPU to perform some task, the peripheral requests the attention of the CPU by generating an interrupt. Ingeneral, the peripheral will not do anything while it is waiting for its interrupt request to be serviced. The peripheral subsystem contains 9 different peripherals all of which operate independent of the other peripherals and the CPU. Four peripherals are used in this thesis. They are the A/D convereter, the General Purpose Timer Units, the Asynchronous Serial Channel, and the CAN-Module.

The C167CRs that make up the breadboard facilities CAN come in four main varieties.

1. Load Nodes
2. DC/DC Converter Controller Node
3. Energy Management Node
4. Serial to CAN Router Node

The software that controls each of these nodes is made up of a 'mainXYZ.asm'[6] object file that is linked to several other object files that control one of the on chip peripherals. A full list of each node and the software that makes up the node can be found in Appendix B.

The files are assembled togther using a DOS batch file entitled 'compXYZ.bat' where XYZ is a unique alphanumeric identifier for each node. 'CompXYZ.bat' first assembles all of the necessary assembly files. It then proceeds to link these files and locate them, and then turn the output of the locater[7] into an Intel hex formatted file. Intel hex is the file format required by the KitCON-167 board. All Intel hex formatted files end in '.hex'. These files can be downloaded to the KitCON-167 boards via the program 'Flasht.exe'. Download of an Intel hex formatted program to one of the KitCON-167 boards is done by first connecting the KitCON-167 board to the COM1 port of the PC. Then, 'flasht' must be typed and entered from a DOS command prompt in the directory that contains the hex file that should be downloaded. The 'Flasht.exe' program will only work properly if it is in the Windows95 path[8]. 'Flasht.exe' does not work under Windows NT.

A microcontroller differs from a PC in that the microcontroller does not come with a preprogrammed boot ROM or BIOS. The information in the PC's BIOS tells the PC's microprocessor how the microprocessor should communcate with the PC's memory and data busses. This code must be provided by the user to the microcontroller. When the C167CR is first powered on, it starts program execution from memory address 00'0000h. In order for the user's program to execute properly, a branch instruction to the start of the program must be located at memory address 00'0000h. 3.3.2

[6] In 'mainXYZ.asm' the XYZ is a unique alphanumeric identifier. For example, 'main114.asm' is the main file for the main assembly language file for CAN node 1 on the 14V bus.

[7] The locator calls the file 'linker.lnv'. This tells the locater where the Flash memory is located and where the RAM is located. This file is the same for all items on the CAN bus.

[8] The 'PATH' statement appears in both the 'Autoexec.bat' and Autoexec.dos' files in Windows95.

```
startupsec  SECTION CODE        ; codesection that contains reset pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP ; end procedure
startupsec ENDS ; end segment
```

Figure 3.8: C167CR Startup Code

The first instruction that is executed after the initial branch is typically 'DISWDT'. This istruction will disable the on chip watch dog timer. The watch dog timer is a timer that, if not serviced before a specific period of time, will reset the chip. This feature is not needed for the breadboard facility, so it is disabled.

After placing the appropriate branch instruction at memory address 00'000h and disabling the watch dog timer, the next thing that needs to be done is to tell the assembler and the linker about the memory that the C167CR can access. The CAN nodes for this network were made up of Phytec KitCON-167 boards. These kits are built around a CAN enabled Siemens C167CR microcontroller. They contained 256kbytes of on board flash memory, and 64kb of RAM. The memory map can be seen in Figure 3.9.

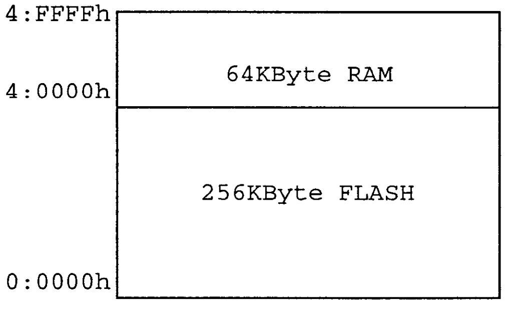


Figure 3.9: Memory Map of Phytec KitCON-167 used in Breadboard Facility

```
 ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
;; End of external memory bus initialization
```

Figure 3.10: Assembly Code that allows External Memory Bus Accesses

```
meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
```

Figure 3.11: Loop Code for C167CR

The C167CR uses it's SYSCON, ADDRSEL and BUSCON registers to control access to off chip memory [6]. Figure 3.3.2 shows the code that would appropriately configure the microcontroller to access the memory on the KitCON-167 boards.

After the memory has been initialized, the 'EINIT' instruction has to be executed. This instruction locks in the memory configuration and allows further code to access the external memory. After this point, the SYSCON, ADDRSEL, and BUSCON registers cannot be changed. Once the 'EINIT' instruction has been executed, the system stack must be configured. After the stack is appropriately configured, each of the on chip peripherals that are to be used can now be configured. Configuration of an on chip peripheral is usually done by calling a function that is located in a different file. This is done as an organizational measure in keep file sizes small and readable. It also improves the abstraction layer between implementation of the software and the interface to that software. This allows the same 'main.asm' file to be used, with very little modification, for all sorts of different programs. Because configuration of most of the on chip peripherals is relatively simple, only the CAN bus initialization will be discussed in this thesis in Section 3.3.2.1.

Once all of the on chip peripherals have been initialized, the CPU must be set perform some sort of continuous loop. The code to do this is shown in Figure 3.3.2. Failure to cause the processor to loop will result in the processor to stop functioning at the end of the function.

3.3.2.1 The CAN Bus

Every CAN message contains 4 main user programmable parts. These parts are

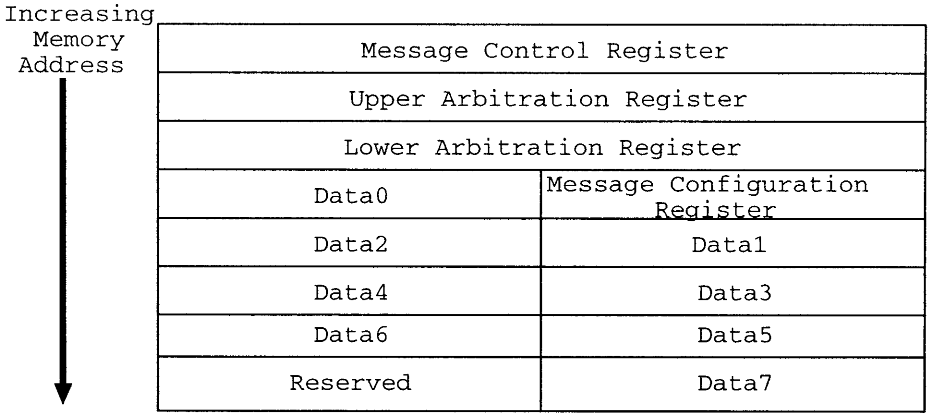
1. Data Length Code
2. Message Direction
3. Arbitration Registers
4. Message Control Registers

Figure 3.12 shows how the major portions of a CAN message are arranged in memory. This grouping of registers in memory is referred to as a Message Object. The C167CR has 15 Message Objects. CAN is capable of transmitting variable length messages of up to 8 bytes in length. It is therefore, necessary to specify within the message, the length of the data field. This is done by setting the Data Length Code value in the Message Configuration Register. Next, each CAN message can either transmit data or receive data. Therefore it is necessary to specify this value by setting the Message Direction bit in the Message Configuration Register. Each CAN message has a unique message ID. This message ID is placed into the Upper Arbitration Register. Message IDs can either be 11 bits in length or they can be 29 bits long. For the purpose of this thesis, 11 bit message IDs have been used. Finally, every CAN message has a Message Control Register that specifies the behavior of the message object with respect to interrupts and how the message object will change when the data fields in the message object change.

3.3.3 Load Nodes

The load nodes were configured to be able to independently turn on and turn off multiple loads. Most nodes were configured to turn on and turn off 2 different loads, but some were configured to control as many as 3 loads. The nodes were also configured to collect current information provided by a node's smart switch's current sense pin. Each load node is able to report the current of each load and also the state (on or off) of each load when the appropriate command from the CAN bus is received. Table 3.3 and Table 3.4 show the messages[9] for each load and the node that they

[9] These are not actually the CAN message numbers, but they are the contents of the Upper Arbitration Register of a CAN message Object from which the CAN message number is generated. In order to generate the actual CAN message ID, the first nibble in the Upper Arbitration Register would be moved into the lst position and then the entire word would be shifted to the right by one bit.



Message Control Register	
Upper Arbitration Register	
Lower Arbitration Register	
Data0	Message Configuration Register
Data2	Data1
Data4	Data3
Data6	Data5
Reserved	Data7

Figure 3.12: CAN Message Object Regsiters and Memory Locations

appear on. All messages marked Receive are configured to receive two different pieces of data. If the received datum is #000001h then the corresponding smart switch is turned on. If the received datum is #000800h then the corresponding smart switch is turned off.

3.3.4 Energy Management Node

The energy management node serves the purpose of both collecting the data necessary to make decisions involved with energy management, and to actually run the energy management algorithm itself. The algorithm was located on this node because it allowed easy access through memory to the collected data. It could, in fact, be located on any node on the network and the necessary data could be simply transmitted to that node across the network. The energy management algorithm is executed once every second. The last piece of data to be collected is the 42V current and direction information. After this datum is stored, the energy management algorithm function is called. The energy management algorithm produces an 8-bit pattern and sends this information across the network to the DC/DC converter node.

The energy management node is configured to collect voltage, current magnitude, current direction, and temperature for each of the batteries. The hardware necessary to collect battery temperature information was not implemented, so the software was written to collect, but ignore, the datum that the A/D collects when it is supposed to collect information about temperature. In total, this board has 6 A/D channels. Each channel is accessed once a second.

Breadboard Loads		
14v Bus Node 1		
CAN Message	CAN Message Direction	CAN Message Number
Power Door Locks	Receive	#0001h
Seat & Door Module	Receive	#2001h
Power Door Locks Current	Transmit	#6001h
Seat & Door Current	Transmit	#4001h
Power Door State	Transmit	#0010h
Seat & Door State	Transmit	#0011h
14v Bus Node 2		
Turn Lights	Receive	#8001h
Turn Lights Current	Transmit	#4007h
Turn Lights State	Transmit	#0012h
14v Bus Node 3		
ABS	Receive	#C001h
Brake Loads	Receive	#E001h
ABS Current	Transmit	#E002h
Brake Loads Current	Transmit	#0002h
ABS State	Transmit	#0013h
Brake State	Transmit	#0014h
Bus Bridge	Receive	#0022h
Bus Bridge Current	Transmit	#0023h
Bus Bridge State	Transmit	#0024h

Table 3.3: 14v Bus CAN Messages

The data is collected as the lower 10-bits of a word of memory. These 10-bits, however, represent a voltage from 0V to 5V not a current of up to 100 amps or a voltage of up to 60 volts. In order to properly use the information, it must be scaled. In the case of the voltage, it is not scaled on the microcontroller, instead, it is scaled and displayed in LabView. This is done because LabView takes care of much of the difficulty of using floating point numbers. In the case of the current, however, because the state of charge of each battery is calculated by integrating the total charge that has entered and exited each battery, it must be scaled on chip. The problem with scaling the measured number is that it could result in a loss of accuracy. This is undesirable, so instead of scaling the measured reading, the initial charge on each battery was scaled before assembling the code, and that scaled number is added to and subtrated from to compute the state of charge for each battery.

The scaling for the initial state of charge for each battery was done as follows. First, the reserve capactiy of the battery is multiplied by 15[11] in order to compute the number of seconds that the battery can be discharged at 100 amps. Then, it must be realized that when the A/D converter produces the 10-bit pattern #03FFh it is actually reading 100 amps of current. If the current is measured every second, then the 10-bit pattern produced by the A/D converter is not only the current, but, by definition, it is also the total charge for one second. Multiplying #03FFh by the number of seconds that the battery can be discharged at 100 amps, returns the state of charge of the battery in a format that the output of the A/D can now be simply added and subtracted from with out any sort of conversion or loss of precision.

The initial value for the 36V battery was #01063E6h and the value for the 12V battery was #02576A0h. These two numbers are both larger than would be allowed by the 16-bit registers of the C167CR, so they are broken into two different words (a high word and a low word) and stored in two different variables in memory. The 10-bit output of the A/D converter is then added to the low word of the battery's state of charge, and, immediately afterward, zero and the carry bit is added to the upper word by using the add-carry instruction. These instructions are executed consecutively as atomic instructions so that they may not be interrupted inbetween and the carry bit be corrupted.

3.3.5 Serial to CAN Router Node

One of the goals of the breadboard facility was to try to explore possible useful functions of having an in-car automobile network. One possible benefit of the network would be in the area of self diagnostics. In the automobile of the future, because loads will be controlled by a digital network and connected off of a power bus, it will be much more difficult to tell where the fault in the network has occurred unless there were some catastrophic failure which left smoke, soot or other physical indicators that clearly indicate the culprit. In the absence of such physical indicators, it might be impossible to track down the fault unless the network has some intelligence and can tell the operator where the fault occurred. It is, therefore, necessary to be able to quickly and easily connect to the in-car network. If it were possible to interface to the in-car CAN network through a serial port, almost any device with a serial port[12] could be programmed to act as diagnostic

[11] The multiple 15 is obtained because the reserve capacity of a battery is the number of minutes that a battery can be discharged at 25 amps. Multiply reserve capacity by 60 and the total number of seconds that the battery can be discharged at 25 amps is known. Divide this new number by 4 and the number of seconds that the battery can be discharged at 100 amps is known.

[12] Serial port in this case means an RS232 port

equipment for the automobile. Therefore, a serial to CAN router was written. This router employs time out error checking and checksum error checking.

In order to be able to translate between CAN and serial, it is necessary to develop rules that will convert a CAN message to a serial message. It is, therefore, necessary to understand the different parts of a CAN message that would come into play in such a translation. Section 3.3.2.1 discusses these parts in detail, but quickly below are the major user programmable parts.

1. Data Length Code
2. Message Direction
3. Arbitration Registers
4. Message Control Registers

The data necessary for each of these parts must be transmitted in the messages going from the PC to the Serial to CAN Router Node. They must then be moved into a CAN message object and transmitted onto the CAN bus. If the serial message sent is simply a command to turn something on the CAN bus on or off, the serial message is put into message object 1. If the message sent from the PC is a request for data, then message object 2 is used. The format of the serial message can be seen in Figure 3.13.

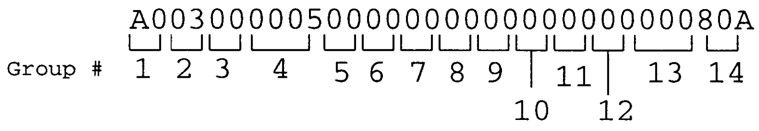


Figure 3.13: Format of Serial Message

All numbers and characters in Figure 3.13 are written in hexidecimal notation. Each character in the message represents a nibble[13] of information. These bytes can be grouped into words or double words. Groups 1 and 14 represent the message delimiters. These are used to prevent LabView

[13] A nibble is defined here as 4 bits.

from removing any leading edge zeros and thereby change the message length. These are not used in computing the checksum of the message. Group 2 represents the data length code. It has a data range of 0h to 8h. Group 3 represents the direction of transmission. It can have the value of either 8h for a transmit message or 0h for a receive message. Group 4 represents the the value that will be placed into the Upper Arbitration Register of the message object. From this value the actual message id of the CAN message can be obtained. Groups 5 through 12 represent the data bytes, but because of how the CAN router is written, only data in groups 6 and 7 will be transmitted, and they will be transmitted as one word with group 6 being the upper byte of the word. The value of #0800h in the 6/7 combination word indicates the the receiving node is to turn off a device, and the value #0001h in the 6/7 combination word indications that the receiving node is to turn on a device. Finally, group 13 represents the checksum of the message. The checksum is computed by simply adding up the values in groups 2, 3, 4, 6, and 7 on a byte by byte basis.

3.3.6 Data Collection Module

The data collection node was designed to prepare the batteries' voltages and currents so that the information could be converted from analog to digital and then used by the energy management algorithm. The information was converted from analog to digital via the Siemen's C167CR on chip 10-bit analog to digital converter. [7] The module was configured to measure voltage, current, and temperature for each battery; however, temperature was not used for this thesis. Because the A/D on the C167CR only has an input range of zero to five volts, all measured signals had to be preprocessed in get them within that range. The 36V battery voltage was measured by dividing the 36V battery's voltage by 11 and then reading that value. The 12V battery's voltage was measured by dividing its voltage by 5 and then reading that value. The current on each battery was measured by passing half the current for each bus through different hall effect current sensors. These sensors returned a current that was $\frac{1}{1000}$ times the sensed current. This current was sent through a 50Ω resistor. This voltage, however, could be either positive or negative, so its absolute value was taken by the circuit in Figure 3.14. This circuit returned both the absolute value of the input, and it returned whether the current was into or out of the battery. If there was 5V on the "Current Direction" terminal, then the current was leaving the battery and if there was 0V on the "Current Direction" terminal then the current was entering the battery. A value of zero at the output of the current direction means that the battery is charging and a value of one at the output of the current direction means that the battery is discharging.

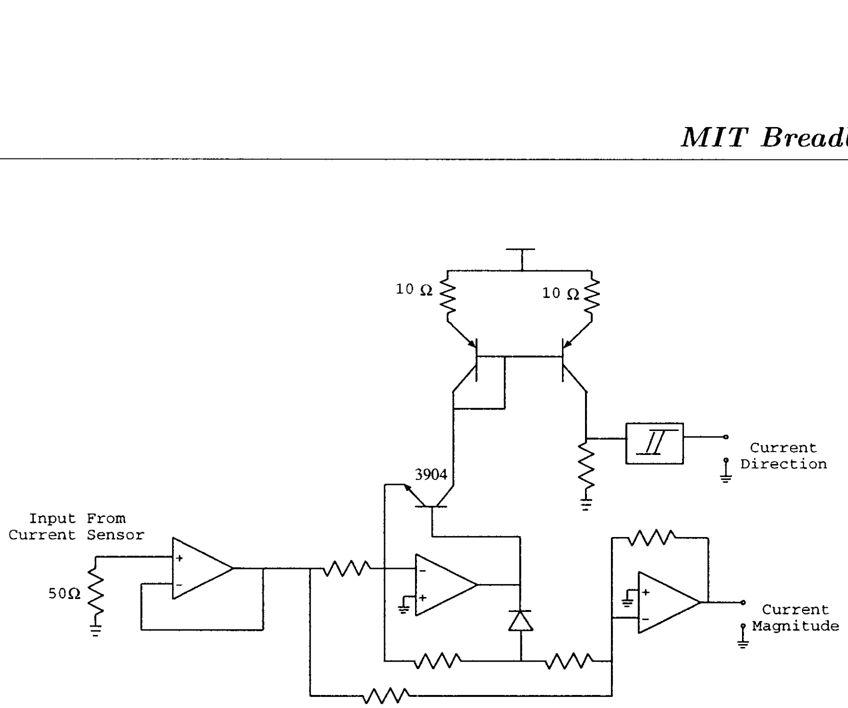


Figure 3.14: Precision Absolute Value Circuit with Direction SubCircuit

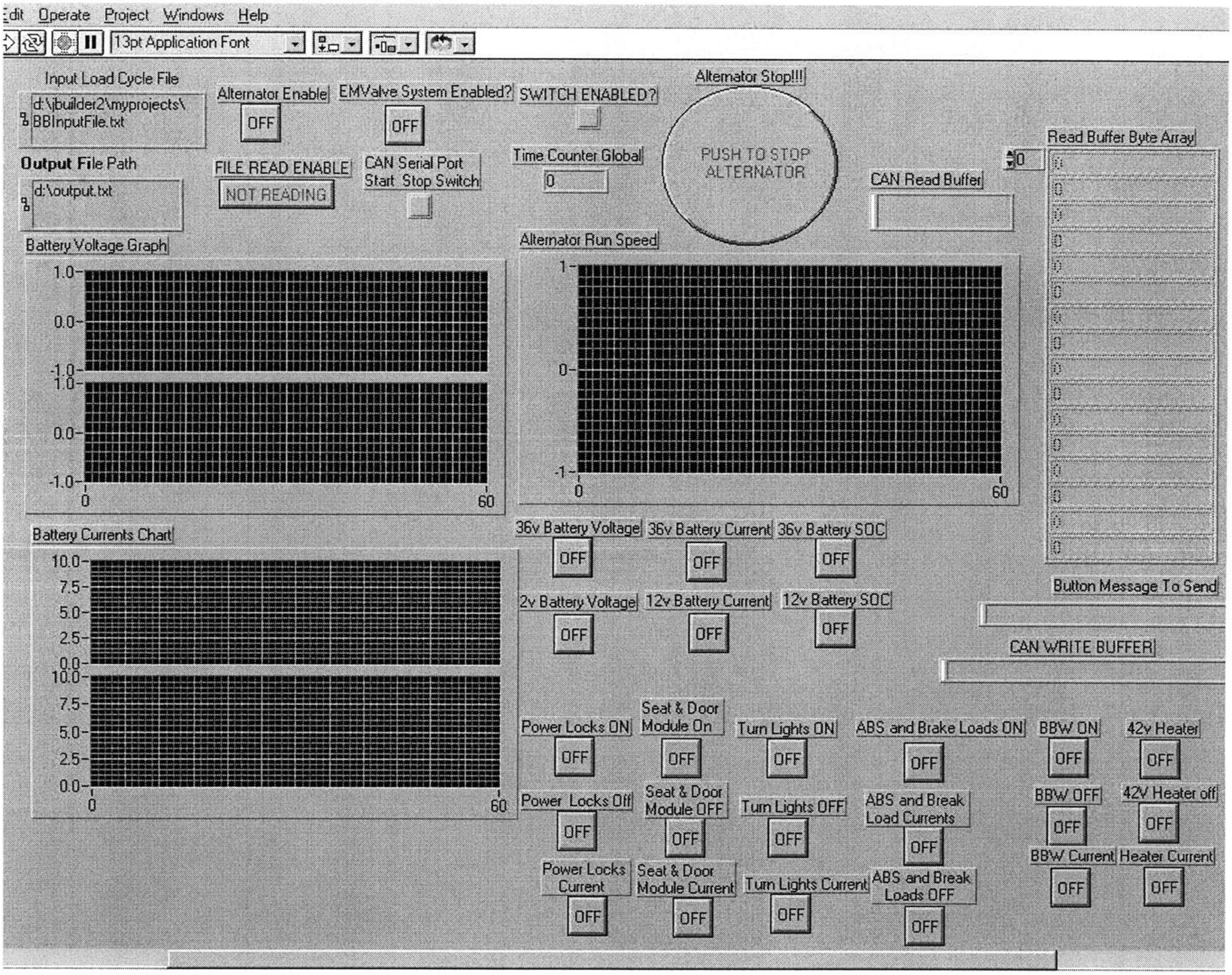
3.3.7 PC Input Files

One goal of the breadboard facility was to be able to allow tests that were run on Saber to be confirmed on the breadboard. The Saber simulations study "the effects of varying vehicle driving speeds and load events on power flow and energy usage [in order] to provide insite into the sizing of key power supply components such as the alternator, batteries, and DC/DC converter" [8]. In order to allow this, a program was written that would take in Saber formatted drive cycles and Saber formatted load cycles and convert them into a tab delimited format that could be read in by the breadboard facility. A copy of the first few lines of a breadboard input file can be seen in Figure 3.3.1.1. The program also takes in a list of the loads that are available on the breadboard facility and those loads' respective CAN Message ID's[14]

[14] CAN Message ID here refers to the value that is loaded into the Upper Arbitration Register of a CAN message object on a Siemens C167CR microcontroller. The actual Message ID can be derived from this value.

Breadboard Loads		
CAN Message	CAN Message Direction	CAN Message Number[10]
42V Bus Node 1		
Brake by Wire	Receive	#0003h
Heated Rear Windows	Receive	#4003h
Brake by Wire Current	Transmit	#6003h
Heated Rear Windows Current	Transmit	#2003h
Brake by Wire State	Transmit	#0015h
Heated Rear Window State	Transmit	#0016h
42v Bus Node 2		
Heater	Receive	#8003h
Rear Seat Heater	Receive	#A003h
Heater Current	Transmit	#C003h
Rear Seat Heater Current	Transmit	#0019h
Heater State	Transmit	#0017h
Rear Seat Heater State	Transmit	#0018h
42v Bus Node 3		
Emissions Air Pump	Receive	#0004h
Heated Windshield	Receive	#4004h
Emissions Air Pump Current	Transmit	#2004h
Heated Windshield Current	Transmit	#6004h
Emmissions Air Pump State	Transmit	#0020h
Heated Windshield State	Transmit	#001Ah
DC/DC Converter Node		
DC/DC Converter Digital Input	Receive	#000Eh
DC/DC Converter Input State	Transmit	#000Fh
DC/DC Converter ON/OFF	Receive	#0021h
Data Collection Node		
42v Voltage	Transmit	#0005h
42v Current & Direction	Transmit	#0006h
42v Temperature	Transmit	#0007h
42v State of Charge	Transmit	#0008h
14v Voltage	Transmit	#0009h
14v Current & Direction	Transmit	#00BAh
14v Temperature	Transmit	#000Bh
14v State of Charge	Transmit	#000Ch

Table 3.4: 42v Bus CAN Messages



Input Load Cycle File: d:\jbuilder2\myprojects\BBInputFile.txt

Output File Path: d:\output.txt

FILE READ ENABLE: NOT READING

Time Counter Global: 0

Figure 3.15: The LabView Breadboard Interface

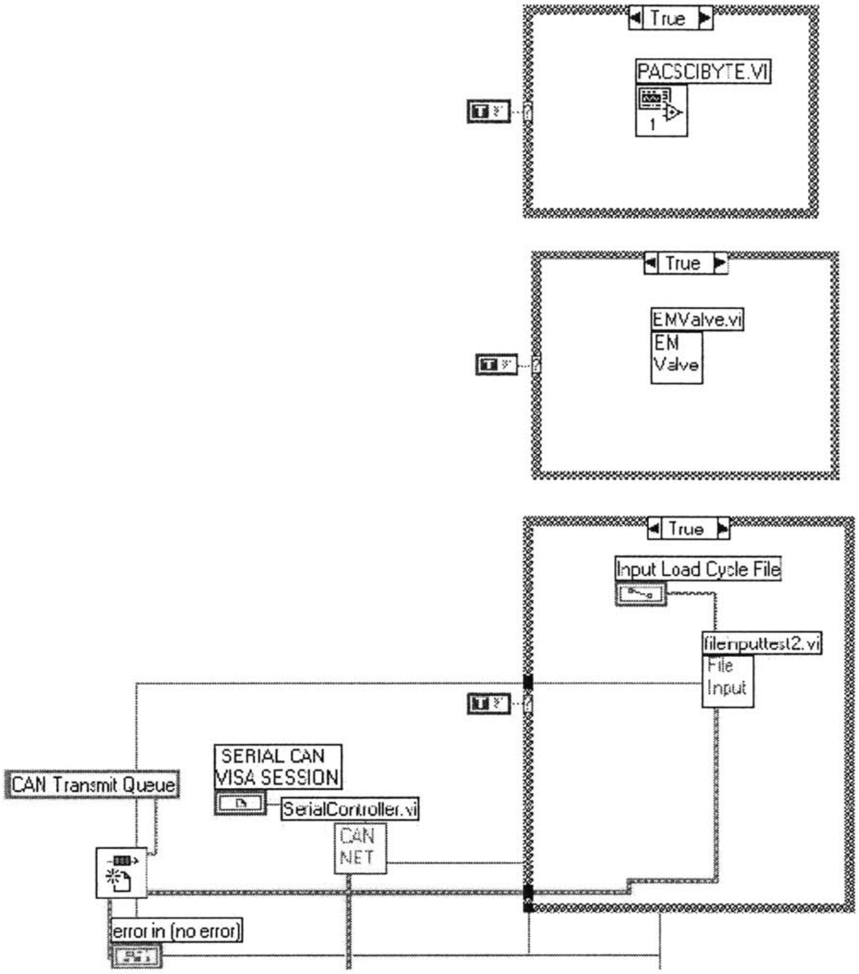


Figure 3.16: The major communicating subsystems

Chapter 4

Test Procedure

This chapter presents the test procedure which was used to measure the effectiveness of the battery voltage regulation energy management algorithm. Testing an energy management algorithm is a 6 stage process. These stages are listed below.

1. Design an energy management algorithm
2. Select a drivecycle to use with it
3. Design an appropriate electrical loadcycle for the selected drivecycle
4. Convert the drivecycle and loadcycle into a breadboard input file
5. Run the breadboard input file on the breadboard test facility
6. Analyze collected data

4.1 Design an Energy Management Algorithm

Energy management algorithm design and implementation is discussed in detail in Chapter 2 of this thesis.

4.1.1 Selecting a Drivecycles

A drivecycle is a data file which contains time, car velocity, and car gear in three columns. The drivecycle's information can be converted to alternator shaft speed using the Eqution 4.1 [8], or engine shaft speed by using Equation 4.2.

$$\text{Alternator Shaft Speed} = v * \frac{10}{36} * \frac{60}{\pi} * d * g_d * g_t * g_{e,a} \tag{4.1}$$

$$\text{Alternator Shaft Speed} = v * \frac{10}{36} * \frac{60}{\pi} * d * g_d * g_t \tag{4.2}$$

The program that generates the breadboard input files actually calculates the engine shaft speed because it actaully controls the speed of the motor that drives the alternator, and that is connected to the alternator at a gearing of 3 to 1.

Variables Used in Car Velocity to Alternator Conversion		
Variable	Description	Ratio
v	Vehicle Driving Speed [km/hr]	
d	Diameter of Vehicle's Tires [m]	0.594
g_d	Differential Gear Ratio	4.0
g_t	Transmission Gear Ratio	
	- Neutral	0
	- 1^{st} Gear	3.071
	- 2^{nd} Gear	1.773
	- 3^{rd} Gear	1.194
	- 4^{th} Gear	0.868
	- 5^{th} Gear	0.700
$g_{e,a}$	Engine-Alternator Gear Ratio	3.0

Table 4.1: Variables Used in Car Velocity to Alternator Conversion

4.1.2 Loadcycles

An electrical loadcycle is a Saber *.scs input file that lists items by name, and lists those item's on and off times. The electrical loadcycle that was used with drivecycle "ece15.dat" was "winter worst ece15". The set of loads that was used for the test can be found in "breadboardloads.txt". Both "winter worst ece15" and "breadboardloads.txt" can be found in Appendix B.12

Drivecycle ece15.dat was selected because it has been tested and shown to work with SABER. As more drivecycles are proven to work with SABER, more will be used. It is the hope that algorithms can be tested on SABER and then verified using the breadboard system. Drivecycle ece15.dat will be matched with a slightly modified version of the electrical loadcycle "winter worst ece15". This electrical loadcycle was used by research unit number six and can be found at the end of this paper.

The goal of this test procedure is to allow the energy management algorithms to be tested on both a computer running Saber and on the MIT breadboard facility. Because the breadboard runs in real time, the hope is that the computer will help eliminate algorithms which don't make any sense and thus save time.

The tests will concentrate on the first two levels of sophistication. The third level will be investigated as part of future research. There will be two rounds of tests. The first series of tests will run using the 14-Volt Bus Regulation algorithm. This is the simplest algorithm and the easiest and cheapest to implement. The results of tests run using this algorithm will be used as a reference to measure the relative performance of the more sophisticated algorithms. The second series of tests will run using the Battery Model level algorithm. The results of these tests will be compared to the results from the 14-Volt Bus Regulation tests.

4.2 Test Procedure

1. Obtain LabView loadcycle.

   This can be obtained by writing one from scratch or by
   translating a SABER drivecycle and loadcycle.

2. Determine number of times to run LabView loadcycle and enter value into LabView.
3. Power on breadboard facility.
4. Start Simulation.
5. Wait until all test runs have been completed.
6. Collect and analyze data.
7. Wait 24 hours and collect battery SOC data.

The data to be collected is

- Open circuit battery voltage before test
- Battery Voltages during test
- Open circuit Battery voltages after test

Chapter 5

Results and Conclusion

Tests were run and data was collected. The open circuit battery voltages before the tests were 36.51 volts and 13.82 volts. The final voltages for each battery (after 10 minutes of rest) were 36.19 volts and 13.22 volts. A plot of battery voltage against time during the test is shown below.

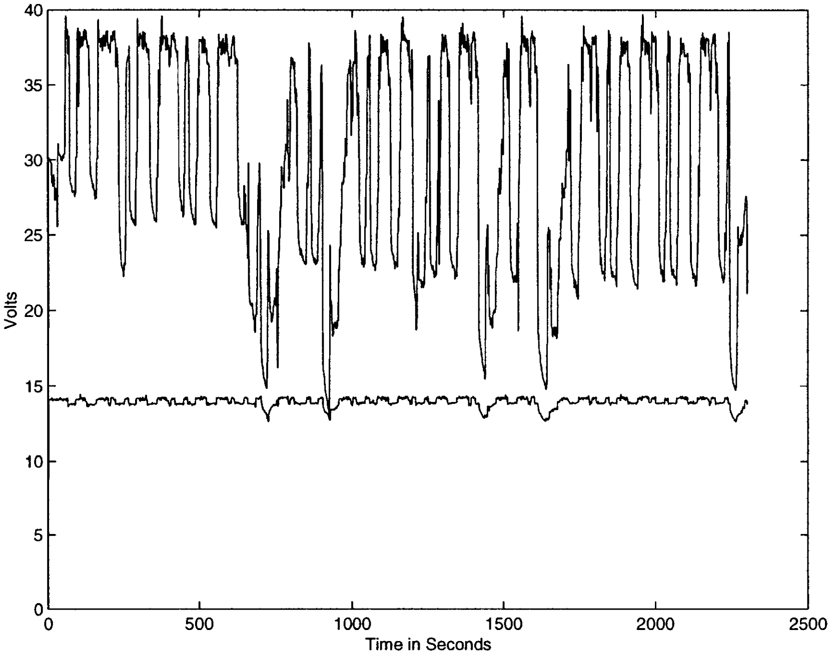


Figure 5.1: Battery Voltages vs Time

It is apparent from Figure 5.1 that the 36V battery's voltage vaired widely while the 12V battery was regulated to a very smooth voltage. This seems to indicate that the 36V battery was supplying the 12V battery a considerable amount of power. This is one of the major flaws of the voltage regulation method of energy management. A more intelligent algorithm would be able to reduce the amount of current demanded by the DC/DC converter. That would have the effect of reducing the 14V bus, but it would also have the effect of reducing some of the ripple in the 42V bus. Although an advanced algorithm was designed and implemented for this thesis, there was not enough time to actually test it, so its results have not been included with the thesis.

The above data shows that the present system of simply regulating the voltage on each battery will probably no longer be adequate in the the 42V/14V dual voltage environment. It will, therefore, be helpful to further invesitgate energy management algorithms.

Appendix A

Complete Sophisticated Energy Management Algorithm

This algorithm was designed and implemented in software in the file ema.asm; however, because of time constraints, it was impossible to fully test it. The table can be read as follows. (12V SOC Region, 36V SOC Region). Negative battery current means that the batteries are draining.

SOC Region	12v Battery Current Sign	36v Battery Current Sign	DC/DC Converter Output
(1,1)	-	-	NONE
(1,1)	-	+	FULL
(1,1)	+	-	OFF
(1,1)	+	+	OFF
(1,2)	-	-	NONE
(1,2)	-	+	UP
(1,2)	+	-	OFF
(1,2)	+	+	OFF
(1,3)	-	-	OFF
(1,3)	-	+	NONE
(1,3)	+	-	OFF
(1,3)	+	+	OFF
(1,4)	-	-	OFF
(1,4)	-	+	OFF
(1,4)	+	-	OFF
(1,4)	+	+	OFF
(1,5)	-	-	OFF
(1,5)	-	+	OFF
(1,5)	+	-	OFF
(1,5)	+	+	OFF

Figure A.1: Decisions made when 12v Battery is in the "Dangerous Overcharge" Region

SOC Region	12v Battery Current Sign	36v Battery Current Sign	DC/DC Converter Output
(2,1)	-	-	UP
(2,1)	-	+	FULL
(2,1)	+	-	NONE
(2,1)	+	+	FULL
(2,2)	-	-	NONE
(2,2)	-	+	UP
(2,2)	+	-	DOWN
(2,2)	+	+	OFF
(2,3)	-	-	NONE
(2,3)	-	+	NONE
(2,3)	+	-	DOWN
(2,3)	+	+	DOWN
(2,4)	-	-	OFF
(2,4)	-	+	OFF
(2,4)	+	-	OFF
(2,4)	+	+	OFF
(2,5)	-	-	OFF
(2,5)	-	+	OFF
(2,5)	+	-	OFF
(2,5)	+	+	OFF

Figure A.2: Decisions made when 12v Battery is in the “Acceptable Overcharge” Region

SOC Region	12v Battery Current Sign	36v Battery Current Sign	DC/DC Converter Output
(3,1)	-	-	FULL
(3,1)	-	+	FULL
(3,1)	+	-	FULL
(3,1)	+	+	FULL
(3,2)	-	-	FULL
(3,2)	-	+	FULL
(3,2)	+	-	FULL
(3,2)	+	+	FULL
(3,3)	-	-	NONE
(3,3)	-	+	NONE
(3,3)	+	-	NONE
(3,3)	+	+	NONE
(3,4)	-	-	DOWN
(3,4)	-	+	DOWN
(3,4)	+	-	DOWN
(3,4)	+	+	DOWN
(3,5)	-	-	OFF
(3,5)	-	+	OFF
(3,5)	+	-	OFF
(3,5)	+	+	OFF

Figure A.3: Decisions made when 12v Battery is in the "Ideal Operation" Region

SOC Region	12v Battery Current Sign	36v Battery Current Sign	DC/DC Converter Output
(4,1)	-	-	FULL
(4,1)	-	+	FULL
(4,1)	+	-	FULL
(4,1)	+	+	FULL
(4,2)	-	-	FULL
(4,2)	-	+	FULL
(4,2)	+	-	FULL
(4,2)	+	+	FULL
(4,3)	-	-	UP
(4,3)	-	+	UP
(4,3)	+	-	UP
(4,3)	+	+	UP
(4,4)	-	-	DOWN
(4,4)	-	+	UP
(4,4)	+	-	DOWN
(4,4)	+	+	NONE
(4,5)	-	-	OFF
(4,5)	-	+	UP
(4,5)	+	-	OFF
(4,5)	+	+	OFF

Figure A.4: Decisions made when 12v Battery is in the “Acceptable Undercharge” Region

SOC Region	12v Battery Current Sign	36v Battery Current Sign	DC/DC Converter Output
(5,1)	-	-	UP
(5,1)	-	+	UP
(5,1)	+	-	UP
(5,1)	+	+	UP
(5,2)	-	-	UP
(5,2)	-	+	UP
(5,2)	+	-	UP
(5,2)	+	+	UP
(5,3)	-	-	UP
(5,3)	-	+	UP
(5,3)	+	-	UP
(5,3)	+	+	UP
(5,4)	-	-	DOWN
(5,4)	-	+	NONE
(5,4)	+	-	DOWN
(5,4)	+	+	NONE
(5,5)	-	-	OFF
(5,5)	-	+	OFF
(5,5)	+	-	OFF
(5,5)	+	+	OFF

Figure A.5: Decisions made when 12v Battery is in the "Dire Undercharge" Region

Appendix B

Breadboard Code

B.1 Organization

This appendix contains the complete code for all items used in the bread board facility.

1. 14V Bus CAN Node 1 B.2
2. 14V Bus CAN Node 2 B.3
3. 14V Bus CAN Node 3 B.4
4. 42V Bus CAN Node 1 B.5
5. 42V Bus CAN Node 2 B.6
6. 42V Bus CAN Node 3 B.7
7. CAN Router B.8
8. Data Acquisition Node B.8
9. DC/DC Converter Node B.10
10. Saber to Breadboard Converter Code B.11
11. Breadboard Loads B.12

B.2 14V Bus CAN Node 1

On the next page starts the code for the 14V bus CAN node 1. The files for the node are as follows.

1. comp112.bat

2. main112.asm
3. cnmod112.asm
4. canmo112.asm
5. cnint112.asm
6. atod112.asm
7. tmrs112.asm
8. linker.lnv
9. Reg167b.def

comp112.bat

```
a166 main112.asm
a166 cnmod112.asm
a166 canmo112.asm
a166 cnint112.asm
a166 atod112.asm
a166 tmrs112.asm
l166 LINK main112.obj cnmod112.obj canmo112.obj cnint112.obj atod112.obj tmrs112.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main112.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
              CALL canin        ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; E0FF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
        RET
canin   ENDP

setall PROC FAR                 ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15             ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1          ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h            ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
      AND R3, ZEROS              ; set R3 to
      OR R3, #00001h            ; message id for message object 1
        MOV [R2],R3              ; message id = #0003h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS       ; fill the Data of the MO with Zeros

        ;; Initialize Message Object 2
        MOV R2, #MCR_M2          ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #5599h            ; RECEIVE INTERRUPT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #02001h           ; The number is the Message ID for Message Object 2
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 000h into first data byte and set to receive
        MOV MCD_M2,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS       ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        MOV R2, #MCR_M3          ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h            ; Generate a receive interrupt if this message object ac
tivates
```

```
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #06001h           ; The number is the Message ID for Message Object 3
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS       ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 4
        MOV R2, #MCR_M4          ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #04001h           ; The number is the Message ID for Message Object 4
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M4,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M4, ZEROS       ; fill the data of the MO with ZEROS



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5          ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #00010h           ; The number is the Message ID for Message Object 5
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS       ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 6
        MOV R2, #MCR_M6          ; start of Message Object 6
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #00011h           ; The number is the Message ID for Message Object 6
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M6,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
```

```
        MOV DATA_M6, ZEROS      ; fill the data of the M0 with ZEROS

        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID         ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h          ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0

exit_function:
        MOV R2,         #0EFFFh

        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A221h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                     ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH MDH
        PUSH MDL

        MOV R2, ADDAT
        MOV R0, R2                      ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2                      ; This is so we can isolate the A/D data and then sc
ale it by

        ;; This code scales the data from the A/D by 21 to get the actual current fl
owing through the BTS550P
        AND R3, #003FFh ; This isolates the lower ten bits of the A/D's output
        MOV R4, #01h    ; There is no scaling done on the controller


        AND R0, #0F000h ;  The channel information is located in the upper nibble
      CMP R0, #01000h   ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, Rear_Seat_Heater_current


        MOV R0, #05555h                 ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0                  ; Kill the Message Control Register

        ;; This gets the actual current value
        MUL R3, R4                      ; The output goes entirely into MDL
        NOP
        MOV DATA_M3, MDL                ; Move the actual current value from the MDL registe
r into the CAN message object
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Rear_Seat_Heater_current:

        MOV R0, #05555h                 ; This bit pattern deactives MCRs
        MOV R1, MCR_M4                  ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0                  ; Kill the Message Control Register
        ;; This code tells me when I have completed a conversion on both channels
        ;; If the leds on port 2 are not counting then You know that the system isn'
t performing conversionsS
        MOV R0, #04h    ;test code
        ADD P2, R0                      ;test code

        ;; This generates the acutal current value
        MUL R3, R4                      ; The output goes entirely into MDL
        NOP
        MOV DATA_M4, MDL                ; for testing purposes
        MOV MCR_M4, R1

exit_routine:
        POP MDL
        POP MDH
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
atod_function ENDP
atod_handlers ENDS
```

```
END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
LOCATE
locatein.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES(‘RAM’ (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO locate.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0              DEFR     0F038h
PT3              DEFR     0F036h
PT2              DEFR     0F034h
PT1              DEFR     0F032h
PT0              DEFR     0F030h

; Bit names
CC0IO            DEFB     P2.0
CC1IO            DEFB     P2.1
CC2IO            DEFB     P2.2
CC3IO            DEFB     P2.3
CC4IO            DEFB     P2.4
CC5IO            DEFB     P2.5
CC6IO            DEFB     P2.6
CC7IO            DEFB     P2.7
CC8IO            DEFB     P2.8
CC9IO            DEFB     P2.9
CC10IO           DEFB     P2.10
CC11IO           DEFB     P2.11
CC12IO           DEFB     P2.12
CC13IO           DEFB     P2.13
CC14IO           DEFB     P2.14
CC15IO           DEFB     P2.15
EX0IN            LIT      ’CC0IO’
EX1IN            LIT      ’CC1IO’
EX2IN            LIT      ’CC2IO’
EX3IN            LIT      ’CC3IO’

T0IN             DEFB     P3.0
T6OUT            DEFB     P3.1
CAPIN            DEFB     P3.2
T3OUT            DEFB     P3.3
T3EUD            DEFB     P3.4
T2IN             DEFB     P3.7
T3IN             DEFB     P3.6
T4IN             DEFB     P3.5
SSDI             DEFB     P3.8
SSDO             DEFB     P3.9
TXD0             DEFB     P3.10
RXD0             DEFB     P3.11
SSCLK            DEFB     P3.13
CLKOUT           DEFB     P3.15

A16              DEFB     P4.0
A17              DEFB     P4.1
A18              DEFB     P4.2
A19              DEFB     P4.3
A20              DEFB     P4.4
A21              DEFB     P4.5
A22              DEFB     P4.6
A23              DEFB     P4.7

AN0              DEFB     P5.0
AN1              DEFB     P5.1
AN2              DEFB     P5.2
AN3              DEFB     P5.3
AN4              DEFB     P5.4
AN5              DEFB     P5.5
AN6              DEFB     P5.6
AN7              DEFB     P5.7
AN8              DEFB     P5.8
AN9              DEFB     P5.9
AN10             DEFB     P5.10
AN11             DEFB     P5.11
AN12             DEFB     P5.12
AN13             DEFB     P5.13
AN14             DEFB     P5.14
AN15             DEFB     P5.15
T6EUD            LIT      ’AN10’
T5EUD            LIT      ’AN11’
T6IN             LIT      ’AN12’
T5IN             LIT      ’AN13’
T4EUD            LIT      ’AN14’
T2EUD            LIT      ’AN15’

POUT0            DEFB     P7.0
POUT1            DEFB     P7.1
POUT2            DEFB     P7.2
POUT3            DEFB     P7.3
CC28IO           DEFB     P7.4
CC29IO           DEFB     P7.5
CC30IO           DEFB     P7.6
CC31IO           DEFB     P7.7

CC16IO           DEFB     P8.0
CC17IO           DEFB     P8.1
CC18IO           DEFB     P8.2
CC19IO           DEFB     P8.3
CC20IO           DEFB     P8.4
CC21IO           DEFB     P8.5
CC22IO           DEFB     P8.6
CC23IO           DEFB     P8.7


T0M              DEFB     T01CON.3
T0R              DEFB     T01CON.6
T1M              DEFB     T01CON.11
T1R              DEFB     T01CON.14
T7M              DEFB     T78CON.3
T7R              DEFB     T78CON.6
T8M              DEFB     T78CON.11
T8R              DEFB     T78CON.14

ACC0             DEFB     CCM0.3
ACC1             DEFB     CCM0.7
ACC2             DEFB     CCM0.11
ACC3             DEFB     CCM0.15

ACC4             DEFB     CCM1.3
ACC5             DEFB     CCM1.7
ACC6             DEFB     CCM1.11
ACC7             DEFB     CCM1.15

ACC8             DEFB     CCM2.3
ACC9             DEFB     CCM2.7
ACC10            DEFB     CCM2.11
ACC11            DEFB     CCM2.15

ACC12            DEFB     CCM3.3
ACC13            DEFB     CCM3.7
ACC14            DEFB     CCM3.11
ACC15            DEFB     CCM3.15

ACC16            DEFB     CCM4.3
ACC17            DEFB     CCM4.7
ACC18            DEFB     CCM4.11
ACC19            DEFB     CCM4.15

ACC20            DEFB     CCM5.3
ACC21            DEFB     CCM5.7
```

```
ACC22               DEFB     CCM5.11
ACC23               DEFB     CCM5.15

ACC24               DEFB     CCM6.3
ACC25               DEFB     CCM6.7
ACC26               DEFB     CCM6.11
ACC27               DEFB     CCM6.15

ACC28               DEFB     CCM7.3
ACC29               DEFB     CCM7.7
ACC30               DEFB     CCM7.11
ACC31               DEFB     CCM7.15

T2R                 DEFB     T2CON.6
T2UD                DEFB     T2CON.7
T2UDE               DEFB     T2CON.8

T3R                 DEFB     T3CON.6
T3UD                DEFB     T3CON.7
T3UDE               DEFB     T3CON.8
T3OE                DEFB     T3CON.9
T3OTL               DEFB     T3CON.10

T4R                 DEFB     T4CON.6
T4UD                DEFB     T4CON.7
T4UDE               DEFB     T4CON.8

T5R                 DEFB     T5CON.6
T5UD                DEFB     T5CON.7
T5UDE               DEFB     T5CON.8
T5CLR               DEFB     T5CON.14
T5SC                DEFB     T5CON.15

T6R                 DEFB     T6CON.6
T6UD                DEFB     T6CON.7
T6UDE               DEFB     T6CON.8
T6OE                DEFB     T6CON.9
T6OTL               DEFB     T6CON.10
T6SR                DEFB     T6CON.15

T2IE                DEFB     T2IC.6
T2IR                DEFB     T2IC.7
T3IE                DEFB     T3IC.6
T3IR                DEFB     T3IC.7
T4IE                DEFB     T4IC.6
T4IR                DEFB     T4IC.7
T5IE                DEFB     T5IC.6
T5IR                DEFB     T5IC.7
T6IE                DEFB     T6IC.6
T6IR                DEFB     T6IC.7

CRIE                DEFB     CRIC.6
CRIR                DEFB     CRIC.7

S0TIE               DEFB     S0TIC.6
S0TIR               DEFB     S0TIC.7
S0RIE               DEFB     S0RIC.6
S0RIR               DEFB     S0RIC.7
S0EIE               DEFB     S0EIC.6
S0EIR               DEFB     S0EIC.7
S0TBIE              DEFB     S0TBIC.6
S0TBIR              DEFB     S0TBIC.7

SSCTIE              DEFB     SSCTIC.6
SSCTIR              DEFB     SSCTIC.7
```

```
SSCRIE              DEFB     SSCRIC.6
SSCRIR              DEFB     SSCRIC.7
SSCEIE              DEFB     SSCEIC.6
SSCEIR              DEFB     SSCEIC.7
SSCTE               LIT      'SSCTEN'
SSCRE               LIT      'SSCREN'
SSCPE               LIT      'SSCPEN'
SSCBE               LIT      'SSCBEN'


CC0IE               DEFB     CC0IC.6
CC0IR               DEFB     CC0IC.7
CC1IE               DEFB     CC1IC.6
CC1IR               DEFB     CC1IC.7
CC2IE               DEFB     CC2IC.6
CC2IR               DEFB     CC2IC.7
CC3IE               DEFB     CC3IC.6
CC3IR               DEFB     CC3IC.7
CC4IE               DEFB     CC4IC.6
CC4IR               DEFB     CC4IC.7
CC5IE               DEFB     CC5IC.6
CC5IR               DEFB     CC5IC.7
CC6IE               DEFB     CC6IC.6
CC6IR               DEFB     CC6IC.7
CC7IE               DEFB     CC7IC.6
CC7IR               DEFB     CC7IC.7
CC8IE               DEFB     CC8IC.6
CC8IR               DEFB     CC8IC.7
CC9IE               DEFB     CC9IC.6
CC9IR               DEFB     CC9IC.7
CC10IE              DEFB     CC10IC.6
CC10IR              DEFB     CC10IC.7
CC11IE              DEFB     CC11IC.6
CC11IR              DEFB     CC11IC.7
CC12IE              DEFB     CC12IC.6
CC12IR              DEFB     CC12IC.7
CC13IE              DEFB     CC13IC.6
CC13IR              DEFB     CC13IC.7
CC14IE              DEFB     CC14IC.6
CC14IR              DEFB     CC14IC.7
CC15IE              DEFB     CC15IC.6
CC15IR              DEFB     CC15IC.7
CC16IE              DEFB     CC16IC.6
CC16IR              DEFB     CC16IC.7
CC17IE              DEFB     CC17IC.6
CC17IR              DEFB     CC17IC.7
CC18IE              DEFB     CC18IC.6
CC18IR              DEFB     CC18IC.7
CC19IE              DEFB     CC19IC.6
CC19IR              DEFB     CC19IC.7
CC20IE              DEFB     CC20IC.6
CC20IR              DEFB     CC20IC.7
CC21IE              DEFB     CC21IC.6
CC21IR              DEFB     CC21IC.7
CC22IE              DEFB     CC22IC.6
CC22IR              DEFB     CC22IC.7
CC23IE              DEFB     CC23IC.6
CC23IR              DEFB     CC23IC.7
CC24IE              DEFB     CC24IC.6
CC24IR              DEFB     CC24IC.7
CC25IE              DEFB     CC25IC.6
CC25IR              DEFB     CC25IC.7
CC26IE              DEFB     CC26IC.6
CC26IR              DEFB     CC26IC.7
CC27IE              DEFB     CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.3 14V Bus CAN Node 2

On the next page starts the code for the 14V bus CAN node 2. The files for the node are as follows.

1. comp212.bat
2. main212.asm
3. cnmod212.asm
4. canmo212.asm
5. cnint212.asm
6. atod212.asm
7. tmrs212.asm
8. linker.lnv
9. Reg167b.def

comp212.bat

```
a166 main212.asm
a166 cnmod212.asm
a166 canmo212.asm
a166 cnint212.asm
a166 atod212.asm
a166 tmrs212.asm
l166 LINK main212.obj cnmod212.obj canmo212.obj cnint212.obj atod212.obj tmrs212.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main212.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
              CALL canin       ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP            ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; EOFF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
        RET
canin   ENDP

setall PROC FAR                ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1         ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h           ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to
      OR R3, #08001h           ; message id for message object 1
        MOV [R2],R3             ; message id = #0003h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS      ; fill the Data of the MO with Zeros


        MOV R2, #MCR_M3         ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h           ; Generate a receive interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #04077h          ; The number is the Message ID for Message Object 3
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M3, ZEROS      ; Fill the Data of the MO with Zeros



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5         ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h                   ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00012h          ; The number is the Message ID for Message Object 5
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS      ; fill the data of the MO with ZEROS

        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID         ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h          ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0

exit_function:
        MOV R2,         #0EFFFh

        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                          ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A221h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                     ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
```

```
        PUSH R3
        PUSH R4
        MOV R2, ADDAT
        MOV R0, R2              ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2              ; This is so we can isolate the A/D voltage sense va
lue

        ;; This code scales the data from the A/D by 21 to get the actual current fl
owing through the BTS550P
        AND R3, #003FFh ; This isolates the lower ten bits of the A/D's output
        MOV R4, #01h    ; No Scaling on the microcontroller


        AND R0, #0F000h ;  The channel information is located in the upper nibble
        CMP R0, #01000h ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, Rear_Seat_Heater_current


        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0          ; Kill the Message Control Register

        MUL R3, R4              ; This generates the acutal current value
        NOP
        MOV DATA_M3, MDL        ; for real
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Rear_Seat_Heater_current:

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M4          ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0          ; Kill the Message Control Register
        MOV R0, #04h    ;test code
        ADD P2, R0              ;test code

        MUL R3,R4               ; This generates the actual current value
        NOP
        MOV DATA_M4, MDL        ; for testing purposes
        MOV MCR_M4, R1

exit_routine:
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
LOCATE
locatein.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES(‘RAM’ (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO locate.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
```

```
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
```

```
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     'CC0IO'
EX1IN           LIT     'CC1IO'
EX2IN           LIT     'CC2IO'
EX3IN           LIT     'CC3IO'

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
```

```
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     'AN10'
T5EUD           LIT     'AN11'
T6IN            LIT     'AN12'
T5IN            LIT     'AN13'
T4EUD           LIT     'AN14'
T2EUD           LIT     'AN15'

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22             DEFB    CCM5.11
ACC23             DEFB    CCM5.15

ACC24             DEFB    CCM6.3
ACC25             DEFB    CCM6.7
ACC26             DEFB    CCM6.11
ACC27             DEFB    CCM6.15

ACC28             DEFB    CCM7.3
ACC29             DEFB    CCM7.7
ACC30             DEFB    CCM7.11
ACC31             DEFB    CCM7.15

T2R               DEFB    T2CON.6
T2UD              DEFB    T2CON.7
T2UDE             DEFB    T2CON.8

T3R               DEFB    T3CON.6
T3UD              DEFB    T3CON.7
T3UDE             DEFB    T3CON.8
T3OE              DEFB    T3CON.9
T3OTL             DEFB    T3CON.10

T4R               DEFB    T4CON.6
T4UD              DEFB    T4CON.7
T4UDE             DEFB    T4CON.8

T5R               DEFB    T5CON.6
T5UD              DEFB    T5CON.7
T5UDE             DEFB    T5CON.8
T5CLR             DEFB    T5CON.14
T5SC              DEFB    T5CON.15

T6R               DEFB    T6CON.6
T6UD              DEFB    T6CON.7
T6UDE             DEFB    T6CON.8
T6OE              DEFB    T6CON.9
T6OTL             DEFB    T6CON.10
T6SR              DEFB    T6CON.15

T2IE              DEFB    T2IC.6
T2IR              DEFB    T2IC.7
T3IE              DEFB    T3IC.6
T3IR              DEFB    T3IC.7
T4IE              DEFB    T4IC.6
T4IR              DEFB    T4IC.7
T5IE              DEFB    T5IC.6
T5IR              DEFB    T5IC.7
T6IE              DEFB    T6IC.6
T6IR              DEFB    T6IC.7

CRIE              DEFB    CRIC.6
CRIR              DEFB    CRIC.7

S0TIE             DEFB    S0TIC.6
S0TIR             DEFB    S0TIC.7
S0RIE             DEFB    S0RIC.6
S0RIR             DEFB    S0RIC.7
S0EIE             DEFB    S0EIC.6
S0EIR             DEFB    S0EIC.7
S0TBIE            DEFB    S0TBIC.6
S0TBIR            DEFB    S0TBIC.7

SSCTIE            DEFB    SSCTIC.6
SSCTIR            DEFB    SSCTIC.7
```

```
SSCRIE            DEFB    SSCRIC.6
SSCRIR            DEFB    SSCRIC.7
SSCEIE            DEFB    SSCEIC.6
SSCEIR            DEFB    SSCEIC.7
SSCTE             LIT     'SSCTEN'
SSCRE             LIT     'SSCREN'
SSCPE             LIT     'SSCPEN'
SSCBE             LIT     'SSCBEN'


CC0IE             DEFB    CC0IC.6
CC0IR             DEFB    CC0IC.7
CC1IE             DEFB    CC1IC.6
CC1IR             DEFB    CC1IC.7
CC2IE             DEFB    CC2IC.6
CC2IR             DEFB    CC2IC.7
CC3IE             DEFB    CC3IC.6
CC3IR             DEFB    CC3IC.7
CC4IE             DEFB    CC4IC.6
CC4IR             DEFB    CC4IC.7
CC5IE             DEFB    CC5IC.6
CC5IR             DEFB    CC5IC.7
CC6IE             DEFB    CC6IC.6
CC6IR             DEFB    CC6IC.7
CC7IE             DEFB    CC7IC.6
CC7IR             DEFB    CC7IC.7
CC8IE             DEFB    CC8IC.6
CC8IR             DEFB    CC8IC.7
CC9IE             DEFB    CC9IC.6
CC9IR             DEFB    CC9IC.7
CC10IE            DEFB    CC10IC.6
CC10IR            DEFB    CC10IC.7
CC11IE            DEFB    CC11IC.6
CC11IR            DEFB    CC11IC.7
CC12IE            DEFB    CC12IC.6
CC12IR            DEFB    CC12IC.7
CC13IE            DEFB    CC13IC.6
CC13IR            DEFB    CC13IC.7
CC14IE            DEFB    CC14IC.6
CC14IR            DEFB    CC14IC.7
CC15IE            DEFB    CC15IC.6
CC15IR            DEFB    CC15IC.7
CC16IE            DEFB    CC16IC.6
CC16IR            DEFB    CC16IC.7
CC17IE            DEFB    CC17IC.6
CC17IR            DEFB    CC17IC.7
CC18IE            DEFB    CC18IC.6
CC18IR            DEFB    CC18IC.7
CC19IE            DEFB    CC19IC.6
CC19IR            DEFB    CC19IC.7
CC20IE            DEFB    CC20IC.6
CC20IR            DEFB    CC20IC.7
CC21IE            DEFB    CC21IC.6
CC21IR            DEFB    CC21IC.7
CC22IE            DEFB    CC22IC.6
CC22IR            DEFB    CC22IC.7
CC23IE            DEFB    CC23IC.6
CC23IR            DEFB    CC23IC.7
CC24IE            DEFB    CC24IC.6
CC24IR            DEFB    CC24IC.7
CC25IE            DEFB    CC25IC.6
CC25IR            DEFB    CC25IC.7
CC26IE            DEFB    CC26IC.6
CC26IR            DEFB    CC26IC.7
CC27IE            DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.4 14V Bus CAN Node 3

On the next page starts the code for the 14V bus CAN node 3. The files for the node are as follows.

1. comp312.bat
2. main312.asm
3. cnmod312.asm
4. canmo312.asm
5. cnint312.asm
6. atod312.asm
7. tmrs312.asm
8. linker.lnv
9. Reg167b.def

comp312.bat

```
a166 main312.asm
a166 cnmod312.asm
a166 canmo312.asm
a166 cnint312.asm
a166 atod312.asm
a166 tmrs312.asm
l166 LINK main312.obj cnmod312.obj canmo312.obj cnint312.obj atod312.obj tmrs312.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main312.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization



        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
                CALL canin      ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; EOFF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
       EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
```

```
        RET
canin   ENDP

setall PROC FAR                 ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1         ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h           ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to
      OR R3, #0C001h           ; message id for message object 1
        MOV [R2],R3             ; message id = #0003h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS      ; fill the Data of the MO with Zeros

        ;; Initialize Message Object 2
        MOV R2, #MCR_M2         ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #5599h           ; RECEIVE INTERRUPT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0E001h          ; The number is the Message ID for Message Object 2
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 000h into first data byte and set to receive
        MOV MCD_M2,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS      ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        MOV R2, #MCR_M3         ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h           ; Generate a receive interrupt if this message object ac
tivates
```

```
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0E002h          ; The number is the Message ID for Message Object 3
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS      ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 4
        MOV R2, #MCR_M4         ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0002h          ; The number is the Message ID for Message Object 4
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M4,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M4, ZEROS      ; fill the data of the MO with ZEROS



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5         ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00013h          ; The number is the Message ID for Message Object 5
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 6
        MOV R2, #MCR_M6         ; start of Message Object 6
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00014h          ; The number is the Message ID for Message Object 6
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M6,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
```

```
        MOV DATA_M6, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 7
        MOV R2, #MCR_M7         ; start of Message Object 7
        AND R1, ZEROS
        OR R1, #5599h           ;
        MOV [R2],R1     ; set MO7's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00022h          ; The number is the Message ID for Message Object 7
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M7,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M7, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 8
        MOV R2, #MCR_M8         ; start of Message Object 8
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO8's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00023h          ; The number is the Message ID for Message Object 8
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M8,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M8, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 9
        MOV R2, #MCR_M9         ; start of Message Object 9
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO9's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00024h          ; The number is the Message ID for Message Object 9
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M9,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M9, ZEROS      ; fill the data of the MO with ZEROS
POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID          ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h           ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle
        CMPB RL0, #09h           ; See if the interrupt came from M07
        JMP cc_Z, message_seven_interrupt

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch
```

```
        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0
        JMP exit_function

message_seven_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M7, R1
        MOV R0, DATA_M7
        MOV MCR_M7, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M9
        MOV MCR_M9, R1
        MOV DATA_M9, R0

        MOV MCR_M9, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_bridge
        BSET P2.2
        JMP exit_function


turn_off_bridge:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.2
        JMP exit_function

exit_function:
        MOV R2,          #0EFFFh
        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                          ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A222h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                     ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
```

```
        PUSH R3
        PUSH R4
        PUSH MDH
        PUSH MDL
        MOV R2, ADDAT
        MOV R0, R2              ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2
        MOV R4, #01h    ; No Scaling on Microcontroller
        AND R0, #0F000h ;  The channel information is located in the upper nibble
        CMP R0, #01000h ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, break_loads_current
        CMP R0, #02000h ; See if the information is coming from Channel 2 of the A/D
        JMP cc_Z, Voltage_Bridge_current

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0          ; Kill the Message Control Register

        MUL R3, R4
        NOP
        MOV DATA_M3, MDL        ; for real
;       MOV P2, R2              ; for testing purposes
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Break_loads_current:

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M4          ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0          ; Kill the Message Control Register
        MOV R0, #08h    ;test code
        ADD P2, R0              ;test code

        MUL R3,R4
        NOP
        MOV DATA_M4, MDL        ; for testing purposes
        MOV MCR_M4, R1
        JMP exit_routine

Voltage_Bridge_current:
        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M8          ; SAVE the Configuration of the MCR
        MOV MCR_M8, R0          ; Kill the Message Control Register

        MUL R3,R4
        NOP
        MOV DATA_M4, MDL        ; for testing purposes
        MOV MCR_M4, R1
        JMP exit_routine



exit_routine:
        POP MDL
        POP MDH
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
```

```
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

linker.lnv

```
LOCATE
locatein.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES('RAM' (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO locate.out
```

```
;**********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;*********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     'CC0IO'
EX1IN           LIT     'CC1IO'
EX2IN           LIT     'CC2IO'
EX3IN           LIT     'CC3IO'

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
```

```
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     'AN10'
T5EUD           LIT     'AN11'
T6IN            LIT     'AN12'
T5IN            LIT     'AN13'
T4EUD           LIT     'AN14'
T2EUD           LIT     'AN15'

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22           DEFB    CCM5.11
ACC23           DEFB    CCM5.15

ACC24           DEFB    CCM6.3
ACC25           DEFB    CCM6.7
ACC26           DEFB    CCM6.11
ACC27           DEFB    CCM6.15

ACC28           DEFB    CCM7.3
ACC29           DEFB    CCM7.7
ACC30           DEFB    CCM7.11
ACC31           DEFB    CCM7.15

T2R             DEFB    T2CON.6
T2UD            DEFB    T2CON.7
T2UDE           DEFB    T2CON.8

T3R             DEFB    T3CON.6
T3UD            DEFB    T3CON.7
T3UDE           DEFB    T3CON.8
T3OE            DEFB    T3CON.9
T3OTL           DEFB    T3CON.10

T4R             DEFB    T4CON.6
T4UD            DEFB    T4CON.7
T4UDE           DEFB    T4CON.8

T5R             DEFB    T5CON.6
T5UD            DEFB    T5CON.7
T5UDE           DEFB    T5CON.8
T5CLR           DEFB    T5CON.14
T5SC            DEFB    T5CON.15

T6R             DEFB    T6CON.6
T6UD            DEFB    T6CON.7
T6UDE           DEFB    T6CON.8
T6OE            DEFB    T6CON.9
T6OTL           DEFB    T6CON.10
T6SR            DEFB    T6CON.15

T2IE            DEFB    T2IC.6
T2IR            DEFB    T2IC.7
T3IE            DEFB    T3IC.6
T3IR            DEFB    T3IC.7
T4IE            DEFB    T4IC.6
T4IR            DEFB    T4IC.7
T5IE            DEFB    T5IC.6
T5IR            DEFB    T5IC.7
T6IE            DEFB    T6IC.6
T6IR            DEFB    T6IC.7

CRIE            DEFB    CRIC.6
CRIR            DEFB    CRIC.7

S0TIE           DEFB    S0TIC.6
S0TIR           DEFB    S0TIC.7
S0RIE           DEFB    S0RIC.6
S0RIR           DEFB    S0RIC.7
S0EIE           DEFB    S0EIC.6
S0EIR           DEFB    S0EIC.7
S0TBIE          DEFB    S0TBIC.6
S0TBIR          DEFB    S0TBIC.7

SSCTIE          DEFB    SSCTIC.6
SSCTIR          DEFB    SSCTIC.7
```

```
SSCRIE          DEFB    SSCRIC.6
SSCRIR          DEFB    SSCRIC.7
SSCEIE          DEFB    SSCEIC.6
SSCEIR          DEFB    SSCEIC.7
SSCTE           LIT     'SSCTEN'
SSCRE           LIT     'SSCREN'
SSCPE           LIT     'SSCPEN'
SSCBE           LIT     'SSCBEN'


CC0IE           DEFB    CC0IC.6
CC0IR           DEFB    CC0IC.7
CC1IE           DEFB    CC1IC.6
CC1IR           DEFB    CC1IC.7
CC2IE           DEFB    CC2IC.6
CC2IR           DEFB    CC2IC.7
CC3IE           DEFB    CC3IC.6
CC3IR           DEFB    CC3IC.7
CC4IE           DEFB    CC4IC.6
CC4IR           DEFB    CC4IC.7
CC5IE           DEFB    CC5IC.6
CC5IR           DEFB    CC5IC.7
CC6IE           DEFB    CC6IC.6
CC6IR           DEFB    CC6IC.7
CC7IE           DEFB    CC7IC.6
CC7IR           DEFB    CC7IC.7
CC8IE           DEFB    CC8IC.6
CC8IR           DEFB    CC8IC.7
CC9IE           DEFB    CC9IC.6
CC9IR           DEFB    CC9IC.7
CC10IE          DEFB    CC10IC.6
CC10IR          DEFB    CC10IC.7
CC11IE          DEFB    CC11IC.6
CC11IR          DEFB    CC11IC.7
CC12IE          DEFB    CC12IC.6
CC12IR          DEFB    CC12IC.7
CC13IE          DEFB    CC13IC.6
CC13IR          DEFB    CC13IC.7
CC14IE          DEFB    CC14IC.6
CC14IR          DEFB    CC14IC.7
CC15IE          DEFB    CC15IC.6
CC15IR          DEFB    CC15IC.7
CC16IE          DEFB    CC16IC.6
CC16IR          DEFB    CC16IC.7
CC17IE          DEFB    CC17IC.6
CC17IR          DEFB    CC17IC.7
CC18IE          DEFB    CC18IC.6
CC18IR          DEFB    CC18IC.7
CC19IE          DEFB    CC19IC.6
CC19IR          DEFB    CC19IC.7
CC20IE          DEFB    CC20IC.6
CC20IR          DEFB    CC20IC.7
CC21IE          DEFB    CC21IC.6
CC21IR          DEFB    CC21IC.7
CC22IE          DEFB    CC22IC.6
CC22IR          DEFB    CC22IC.7
CC23IE          DEFB    CC23IC.6
CC23IR          DEFB    CC23IC.7
CC24IE          DEFB    CC24IC.6
CC24IR          DEFB    CC24IC.7
CC25IE          DEFB    CC25IC.6
CC25IR          DEFB    CC25IC.7
CC26IE          DEFB    CC26IC.6
CC26IR          DEFB    CC26IC.7
CC27IE          DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.5 42V Bus CAN Node 1

On the next page starts the code for the 42V bus CAN node 1. The files for the node are as follows.

1. comp142.bat
2. main142.asm
3. cnmod142.asm
4. canmo142.asm
5. cnint142.asm
6. atod142.asm
7. tmrs142.asm
8. linker.lnv
9. Reg167b.def

comp142.bat

```
a166 main142.asm
a166 cnmod142.asm
a166 canmo142.asm
a166 cnint142.asm
a166 atod142.asm
a166 tmrs142.asm
l166 LINK main142.obj cnmod142.obj canmo142.obj cnint142.obj atod142.obj tmrs142.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main142.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
                CALL canin      ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; EOFF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
```

```
        RET
canin   ENDP

setall PROC FAR                 ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1          ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h            ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
      AND R3, ZEROS              ; set R3 to
      OR R3, #00003h            ; message id for message object 1
        MOV [R2],R3              ; message id = #0003h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS       ; fill the Data of the MO with Zeros

        ;; Initialize Message Object 2
        MOV R2, #MCR_M2          ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #5599h            ; RECEIVE INTERRUPT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #04003h           ; The number is the Message ID for Message Object 2
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 000h into first data byte and set to receive
        MOV MCD_M2,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS       ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        MOV R2, #MCR_M3          ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h            ; Generate a receive interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #06003h           ; The number is the Message ID for Message Object 3
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS       ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 4
        MOV R2, #MCR_M4          ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #02003h           ; The number is the Message ID for Message Object 4
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M4,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M4, ZEROS       ; fill the data of the MO with ZEROS



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5          ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #00015h           ; The number is the Message ID for Message Object 5
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS       ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 6
        MOV R2, #MCR_M6          ; start of Message Object 6
        AND R1, ZEROS
        OR R1, #5595h            ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #00016h           ; The number is the Message ID for Message Object 6
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to receive
        MOV MCD_M6,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
```

```
        MOV DATA_M6, ZEROS      ; fill the data of the MO with ZEROS

        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID         ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h          ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

 turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

 message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0

exit_function:
        MOV R2,         #0EFFFh

        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A221h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                     ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
```

```
        PUSH R3
        PUSH R4
        PUSH MDH
        PUSH MDL
        MOV R2, ADDAT
        MOV R0, R2              ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2              ; This is so we can isolate the A/D data
        AND R3, #03FFh  ; This isolates the A/D data
        MOV R4, #01h    ;       No Scaling to be done on Microcontroller
        AND R0, #0F000h ;  The channel information is located in the upper nibble
      CMP R0, #01000h   ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, Rear_Seat_Heater_current


        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0          ; Kill the Message Control Register

        ;  This multiplication returns the actual value of the current flowing throu
gh the transistor
        MUL R3, R4
        NOP
        MOV DATA_M3, MDL        ; for real
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Rear_Seat_Heater_current:

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M4          ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0          ; Kill the Message Control Register
        ;; This test code counts out on Port 2 and if it doesn't
        ;; Then that means that the A/D and timer aren't working
        MOV R0, #04h    ;test code
        ADD P2, R0              ;test code

        MUL R3, R4
        NOP
        MOV DATA_M4, MDL        ; for testing purposes
        MOV MCR_M4, R1

exit_routine:
        POP MDL
        POP MDH
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
LOCATE
locatein.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES(’RAM’ (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO locate.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;*********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     'CC0IO'
EX1IN           LIT     'CC1IO'
EX2IN           LIT     'CC2IO'
EX3IN           LIT     'CC3IO'

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     'AN10'
T5EUD           LIT     'AN11'
T6IN            LIT     'AN12'
T5IN            LIT     'AN13'
T4EUD           LIT     'AN14'
T2EUD           LIT     'AN15'

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22               DEFB    CCM5.11
ACC23               DEFB    CCM5.15

ACC24               DEFB    CCM6.3
ACC25               DEFB    CCM6.7
ACC26               DEFB    CCM6.11
ACC27               DEFB    CCM6.15

ACC28               DEFB    CCM7.3
ACC29               DEFB    CCM7.7
ACC30               DEFB    CCM7.11
ACC31               DEFB    CCM7.15

T2R                 DEFB    T2CON.6
T2UD                DEFB    T2CON.7
T2UDE               DEFB    T2CON.8

T3R                 DEFB    T3CON.6
T3UD                DEFB    T3CON.7
T3UDE               DEFB    T3CON.8
T3OE                DEFB    T3CON.9
T3OTL               DEFB    T3CON.10

T4R                 DEFB    T4CON.6
T4UD                DEFB    T4CON.7
T4UDE               DEFB    T4CON.8

T5R                 DEFB    T5CON.6
T5UD                DEFB    T5CON.7
T5UDE               DEFB    T5CON.8
T5CLR               DEFB    T5CON.14
T5SC                DEFB    T5CON.15

T6R                 DEFB    T6CON.6
T6UD                DEFB    T6CON.7
T6UDE               DEFB    T6CON.8
T6OE                DEFB    T6CON.9
T6OTL               DEFB    T6CON.10
T6SR                DEFB    T6CON.15

T2IE                DEFB    T2IC.6
T2IR                DEFB    T2IC.7
T3IE                DEFB    T3IC.6
T3IR                DEFB    T3IC.7
T4IE                DEFB    T4IC.6
T4IR                DEFB    T4IC.7
T5IE                DEFB    T5IC.6
T5IR                DEFB    T5IC.7
T6IE                DEFB    T6IC.6
T6IR                DEFB    T6IC.7

CRIE                DEFB    CRIC.6
CRIR                DEFB    CRIC.7

S0TIE               DEFB    S0TIC.6
S0TIR               DEFB    S0TIC.7
S0RIE               DEFB    S0RIC.6
S0RIR               DEFB    S0RIC.7
S0EIE               DEFB    S0EIC.6
S0EIR               DEFB    S0EIC.7
S0TBIE              DEFB    S0TBIC.6
S0TBIR              DEFB    S0TBIC.7

SSCTIE              DEFB    SSCTIC.6
SSCTIR              DEFB    SSCTIC.7
```

```
SSCRIE              DEFB    SSCRIC.6
SSCRIR              DEFB    SSCRIC.7
SSCEIE              DEFB    SSCEIC.6
SSCEIR              DEFB    SSCEIC.7
SSCTE               LIT     'SSCTEN'
SSCRE               LIT     'SSCREN'
SSCPE               LIT     'SSCPEN'
SSCBE               LIT     'SSCBEN'


CC0IE               DEFB    CC0IC.6
CC0IR               DEFB    CC0IC.7
CC1IE               DEFB    CC1IC.6
CC1IR               DEFB    CC1IC.7
CC2IE               DEFB    CC2IC.6
CC2IR               DEFB    CC2IC.7
CC3IE               DEFB    CC3IC.6
CC3IR               DEFB    CC3IC.7
CC4IE               DEFB    CC4IC.6
CC4IR               DEFB    CC4IC.7
CC5IE               DEFB    CC5IC.6
CC5IR               DEFB    CC5IC.7
CC6IE               DEFB    CC6IC.6
CC6IR               DEFB    CC6IC.7
CC7IE               DEFB    CC7IC.6
CC7IR               DEFB    CC7IC.7
CC8IE               DEFB    CC8IC.6
CC8IR               DEFB    CC8IC.7
CC9IE               DEFB    CC9IC.6
CC9IR               DEFB    CC9IC.7
CC10IE              DEFB    CC10IC.6
CC10IR              DEFB    CC10IC.7
CC11IE              DEFB    CC11IC.6
CC11IR              DEFB    CC11IC.7
CC12IE              DEFB    CC12IC.6
CC12IR              DEFB    CC12IC.7
CC13IE              DEFB    CC13IC.6
CC13IR              DEFB    CC13IC.7
CC14IE              DEFB    CC14IC.6
CC14IR              DEFB    CC14IC.7
CC15IE              DEFB    CC15IC.6
CC15IR              DEFB    CC15IC.7
CC16IE              DEFB    CC16IC.6
CC16IR              DEFB    CC16IC.7
CC17IE              DEFB    CC17IC.6
CC17IR              DEFB    CC17IC.7
CC18IE              DEFB    CC18IC.6
CC18IR              DEFB    CC18IC.7
CC19IE              DEFB    CC19IC.6
CC19IR              DEFB    CC19IC.7
CC20IE              DEFB    CC20IC.6
CC20IR              DEFB    CC20IC.7
CC21IE              DEFB    CC21IC.6
CC21IR              DEFB    CC21IC.7
CC22IE              DEFB    CC22IC.6
CC22IR              DEFB    CC22IC.7
CC23IE              DEFB    CC23IC.6
CC23IR              DEFB    CC23IC.7
CC24IE              DEFB    CC24IC.6
CC24IR              DEFB    CC24IC.7
CC25IE              DEFB    CC25IC.6
CC25IR              DEFB    CC25IC.7
CC26IE              DEFB    CC26IC.6
CC26IR              DEFB    CC26IC.7
CC27IE              DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.6 42V Bus CAN Node 2

On the next page starts the code for the 42V bus CAN node 2. The files for the node are as follows.

1. comp242.bat
2. main242.asm
3. cnmod242.asm
4. canmo242.asm
5. cnint242.asm
6. atod242.asm
7. tmrs242.asm
8. linker.lnv
9. Reg167b.def

```
a166 main242.asm
a166 cnmod242.asm
a166 canmo242.asm
a166 cnint242.asm
a166 atod242.asm
a166 tmrs242.asm
l166 LINK main242.obj cnmod242.obj canmo242.obj cnint242.obj atod242.obj tmrs242.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main242.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
              CALL canin       ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

cnmod242.asm

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15            ; define a common register area of 16 registers
GLOBAL  canin                    ; The function must be declared Global at the
                                 ; beginning of the module

EXTERN  canmocfg:FAR            ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE            ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h           ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h          ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh          ; E0FF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1            ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1            ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1            ; lower mask of last register

        CALL setall              ; sets all of the CAN registers to off

        CALL canmocfg            ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h             ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
        RET
canin   ENDP

setall PROC FAR                 ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h             ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h            ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h            ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4              ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg       ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1         ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h           ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to
      OR R3, #08003h           ; message id for message object 1
        MOV [R2],R3             ; message id = #0003h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS      ; fill the Data of the MO with Zeros

        ;; Initialize Message Object 2
        MOV R2, #MCR_M2         ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #5599h           ; RECEIVE INTERRUPT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0A003h          ; The number is the Message ID for Message Object 2
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h           ; put 000h into first data byte and set to receive
        MOV MCD_M2,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS      ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        MOV R2, #MCR_M3         ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h           ; Generate a receive interrupt if this message object ac
tivates
```

```
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0C003h          ; The number is the Message ID for Message Object 3
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS      ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 4
        MOV R2, #MCR_M4         ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00019h          ; The number is the Message ID for Message Object 4
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M4,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M4, ZEROS      ; fill the data of the MO with ZEROS



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5         ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00017h          ; The number is the Message ID for Message Object 5
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 6
        MOV R2, #MCR_M6         ; start of Message Object 6
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00018h          ; The number is the Message ID for Message Object 6
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M6,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
```

```
        MOV DATA_M6, ZEROS       ; fill the data of the MO with ZEROS

        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID         ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h          ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0

exit_function:
        MOV R2,         #0EFFFh

        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                          ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A221h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                     ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
```

```
        PUSH R3
        PUSH R4
        PUSH MDH
        PUSH MDL

        MOV R2, ADDAT
        MOV R0, R2              ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2              ; This is so we can isolate the DATA on the A/D
        AND R3, #03FFh  ; this isolates the A/D data
        MOV R4, #01h    ;  No scaling on microcontroller
        AND R0, #0F000h ;  The channel information is located in the upper nibble
        CMP R0, #01000h ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, Rear_Seat_Heater_current


        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0          ; Kill the Message Control Register

        MULU R3, R4
        NOP
        MOV DATA_M3, MDL        ; for real
;       MOV P2, R2              ; for testing purposes
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Rear_Seat_Heater_current:

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M4          ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0          ; Kill the Message Control Register
        MOV R0, #04h    ;test code
        ADD P2, R0              ;test code


        MULU R3, R4
        NOP
        MOV DATA_M4, MDL        ; for real
        MOV MCR_M4, R1

exit_routine:
        POP MDL
        POP MDH
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
LOCATE
main.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES('RAM' (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO main.out
```

```
;**********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;*********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
```

```
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     ‘CC0IO’
EX1IN           LIT     ‘CC1IO’
EX2IN           LIT     ‘CC2IO’
EX3IN           LIT     ‘CC3IO’

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
```

```
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     ‘AN10’
T5EUD           LIT     ‘AN11’
T6IN            LIT     ‘AN12’
T5IN            LIT     ‘AN13’
T4EUD           LIT     ‘AN14’
T2EUD           LIT     ‘AN15’

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22           DEFB    CCM5.11
ACC23           DEFB    CCM5.15

ACC24           DEFB    CCM6.3
ACC25           DEFB    CCM6.7
ACC26           DEFB    CCM6.11
ACC27           DEFB    CCM6.15

ACC28           DEFB    CCM7.3
ACC29           DEFB    CCM7.7
ACC30           DEFB    CCM7.11
ACC31           DEFB    CCM7.15

T2R             DEFB    T2CON.6
T2UD            DEFB    T2CON.7
T2UDE           DEFB    T2CON.8

T3R             DEFB    T3CON.6
T3UD            DEFB    T3CON.7
T3UDE           DEFB    T3CON.8
T3OE            DEFB    T3CON.9
T3OTL           DEFB    T3CON.10

T4R             DEFB    T4CON.6
T4UD            DEFB    T4CON.7
T4UDE           DEFB    T4CON.8

T5R             DEFB    T5CON.6
T5UD            DEFB    T5CON.7
T5UDE           DEFB    T5CON.8
T5CLR           DEFB    T5CON.14
T5SC            DEFB    T5CON.15

T6R             DEFB    T6CON.6
T6UD            DEFB    T6CON.7
T6UDE           DEFB    T6CON.8
T6OE            DEFB    T6CON.9
T6OTL           DEFB    T6CON.10
T6SR            DEFB    T6CON.15

T2IE            DEFB    T2IC.6
T2IR            DEFB    T2IC.7
T3IE            DEFB    T3IC.6
T3IR            DEFB    T3IC.7
T4IE            DEFB    T4IC.6
T4IR            DEFB    T4IC.7
T5IE            DEFB    T5IC.6
T5IR            DEFB    T5IC.7
T6IE            DEFB    T6IC.6
T6IR            DEFB    T6IC.7

CRIE            DEFB    CRIC.6
CRIR            DEFB    CRIC.7

S0TIE           DEFB    S0TIC.6
S0TIR           DEFB    S0TIC.7
S0RIE           DEFB    S0RIC.6
S0RIR           DEFB    S0RIC.7
S0EIE           DEFB    S0EIC.6
S0EIR           DEFB    S0EIC.7
S0TBIE          DEFB    S0TBIC.6
S0TBIR          DEFB    S0TBIC.7

SSCTIE          DEFB    SSCTIC.6
SSCTIR          DEFB    SSCTIC.7
```

```
SSCRIE          DEFB    SSCRIC.6
SSCRIR          DEFB    SSCRIC.7
SSCEIE          DEFB    SSCEIC.6
SSCEIR          DEFB    SSCEIC.7
SSCTE           LIT     'SSCTEN'
SSCRE           LIT     'SSCREN'
SSCPE           LIT     'SSCPEN'
SSCBE           LIT     'SSCBEN'


CC0IE           DEFB    CC0IC.6
CC0IR           DEFB    CC0IC.7
CC1IE           DEFB    CC1IC.6
CC1IR           DEFB    CC1IC.7
CC2IE           DEFB    CC2IC.6
CC2IR           DEFB    CC2IC.7
CC3IE           DEFB    CC3IC.6
CC3IR           DEFB    CC3IC.7
CC4IE           DEFB    CC4IC.6
CC4IR           DEFB    CC4IC.7
CC5IE           DEFB    CC5IC.6
CC5IR           DEFB    CC5IC.7
CC6IE           DEFB    CC6IC.6
CC6IR           DEFB    CC6IC.7
CC7IE           DEFB    CC7IC.6
CC7IR           DEFB    CC7IC.7
CC8IE           DEFB    CC8IC.6
CC8IR           DEFB    CC8IC.7
CC9IE           DEFB    CC9IC.6
CC9IR           DEFB    CC9IC.7
CC10IE          DEFB    CC10IC.6
CC10IR          DEFB    CC10IC.7
CC11IE          DEFB    CC11IC.6
CC11IR          DEFB    CC11IC.7
CC12IE          DEFB    CC12IC.6
CC12IR          DEFB    CC12IC.7
CC13IE          DEFB    CC13IC.6
CC13IR          DEFB    CC13IC.7
CC14IE          DEFB    CC14IC.6
CC14IR          DEFB    CC14IC.7
CC15IE          DEFB    CC15IC.6
CC15IR          DEFB    CC15IC.7
CC16IE          DEFB    CC16IC.6
CC16IR          DEFB    CC16IC.7
CC17IE          DEFB    CC17IC.6
CC17IR          DEFB    CC17IC.7
CC18IE          DEFB    CC18IC.6
CC18IR          DEFB    CC18IC.7
CC19IE          DEFB    CC19IC.6
CC19IR          DEFB    CC19IC.7
CC20IE          DEFB    CC20IC.6
CC20IR          DEFB    CC20IC.7
CC21IE          DEFB    CC21IC.6
CC21IR          DEFB    CC21IC.7
CC22IE          DEFB    CC22IC.6
CC22IR          DEFB    CC22IC.7
CC23IE          DEFB    CC23IC.6
CC23IR          DEFB    CC23IC.7
CC24IE          DEFB    CC24IC.6
CC24IR          DEFB    CC24IC.7
CC25IE          DEFB    CC25IC.6
CC25IR          DEFB    CC25IC.7
CC26IE          DEFB    CC26IC.6
CC26IR          DEFB    CC26IC.7
CC27IE          DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.7 42V Bus CAN Node 3

On the next page starts the code for the 42V bus CAN node 3. The files for the node are as follows.

1. comp342.bat
2. main342.asm
3. cnmod342.asm
4. canmo342.asm
5. cnint342.asm
6. atod342.asm
7. tmrs342.asm
8. linker.lnv
9. Reg167b.def

```
a166 main342.asm
a166 cnmod342.asm
a166 canmo342.asm
a166 cnint342.asm
a166 atod342.asm
a166 tmrs342.asm
l166 LINK main342.obj cnmod342.obj canmo342.obj cnint342.obj atod342.obj tmrs342.obj TO
locatein.lno
l166 @linker.lnv
ihex166 -i16 locate.out -o main.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                             ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15               ; define a common register area of 16 register

SSKDEF 4                            ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                    ; Can function
EXTERN atod_initialize:FAR                  ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT              ; disable the watchdog timer
                BSET IEN            ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT               ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h              ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure Port 2 is in push/pull mode
                MOV ODP2, ONES

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization


        ;; Initialize A/D timer
                CALL atod_timer_initialize; timers
        ;; End of A/D timer initialization
```

```
        ;; Initialize CAN Bus
              CALL canin        ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; E0FF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
```

```
        RET
canin   ENDP

setall PROC FAR                ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15             ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1          ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h            ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
      AND R3, ZEROS              ; set R3 to
      OR R3, #00004h            ; message id for message object 1
        MOV [R2],R3              ; message id = #0003h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 0AAh into first data byte and set to receive
      MOV MCD_M1,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS       ; fill the Data of the MO with Zeros

        ;; Initialize Message Object 2
        MOV R2, #MCR_M2          ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #5599h            ; RECEIVE INTERRUPT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R6 to zero
        OR R3, #04004h           ; The number is the Message ID for Message Object 2
        MOV [R2],R3              ; message id = 0
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; put 000h into first data byte and set to receive
        MOV MCD_M2,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS       ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        MOV R2, #MCR_M3          ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h            ; Generate a receive interrupt if this message object ac
tivates
```

```
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #02004h          ; The number is the Message ID for Message Object 3
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 000h into first data byte and set to receive
        MOV MCD_M3,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS      ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 4
        MOV R2, #MCR_M4         ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #06004h          ; The number is the Message ID for Message Object 4
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M4,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M4, ZEROS      ; fill the data of the MO with ZEROS



        ;; Initialize Message Object 5
        MOV R2, #MCR_M5         ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #00020h          ; The number is the Message ID for Message Object 5
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M5,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M5, ZEROS      ; fill the data of the MO with ZEROS


        ;; Initialize Message Object 6
        MOV R2, #MCR_M6         ; start of Message Object 6
        AND R1, ZEROS
        OR R1, #5595h           ;
        MOV [R2],R1     ; set MO4's Control register
        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R6 to zero
        OR R3, #0001Ah          ; The number is the Message ID for Message Object 6
        MOV [R2],R3             ; message id = 0
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to receive
        MOV MCD_M6,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
```

```
        MOV DATA_M6, ZEROS      ; fill the data of the MO with ZEROS

        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2

        MOVB RL0, INTID         ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h          ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV MCR_M2, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2,MCR_M6
        MOV MCR_M6, R1
        MOV DATA_M6, R0
        MOV MCR_M6, R2
        CMP R0, #01h
        JMP cc_NZ, turn_off_heated_rear_window
        BSET P2.1
        JMP exit_function

turn_off_heated_rear_window:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.1
        JMP exit_function

message_one_interrupt:
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2
        ;; Now setup M5 so it can respond to queries about
        ;; the state of the switch

        MOV R2, MCR_M5
        MOV MCR_M5, R1
        MOV DATA_M5, R0

        MOV MCR_M5, R2
        CMP R0, #01h
        JMP cc_NZ, turn_heater_off
        BSET P2.0
        JMP exit_function


turn_heater_off:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        BCLR P2.0

exit_function:
        MOV R2,         #0EFFFh

        AND C1CSR, R2
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_initialize

        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5


atod_setup SECTION CODE

atod_initialize PROC FAR
        ;; Initialize variables

        ;; This below line of code setups up the A/D converter
        ;; for 2 channels and single conversion.
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A221h      ; setup A/D control register


        ;; Set the channel to which the data should be written
        ;; when the first "A/D is done" interrupt occurs

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #006Fh
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                    ORG 0A0H
                    CALL atod_function
                    RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable
        PUSH R0
        PUSH R1
        PUSH R2
```

```
        PUSH R3
        PUSH R4
        PUSH MDH
        PUSH MDL

        MOV R2, ADDAT
        MOV R0, R2              ; This is so we can isolate the A/D channel from whi
ch the data is coming
        MOV R3, R2              ; This is so we can isolate the A/D data
        AND R3, #03FFh  ; This isolates the A/D data
        MOV R4, #01h    ; No scaling on microcontroller
        AND R0, #0F000h ;  The channel information is located in the upper nibble
        CMP R0, #01000h ;  See if the information is coming from Channel 1 of the A/
D
        JMP cc_Z, Heated_Windshield_current


        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M3  ; SAVE the Configuration of the MCR
        MOV MCR_M3, R0          ; Kill the Message Control Register

        MUL R3, R4
        NOP
        MOV DATA_M3, MDL        ; for real
;       MOV P2, R2              ; for testing purposes
        MOV MCR_M3, R1
        BSET T3R
        JMP exit_routine


Heated_Windshield_current:

        MOV R0, #05555h         ; This bit pattern deactives MCRs
        MOV R1, MCR_M4          ; SAVE the Configuration of the MCR
        MOV MCR_M4, R0          ; Kill the Message Control Register
        MOV R0, #04h    ;test code
        ADD P2, R0              ;test code

        MUL R3, R4
        NOP
        MOV DATA_M4, MDL        ; for testing purposes
        MOV MCR_M4, R1

exit_routine:
        POP MDL
        POP MDH
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15

GLOBAL atod_timer_initialize

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        MOV T3CON, #0004h       ; setup Core Timer T3
        MOV T3IC, #002Bh
        MOV T3, #0000h          ; Make the value in the counter equal to zero
        BSET T3IE               ; enable the timer interrupt
        BSET T3R                ; start the timer
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP

atod_timer_handler PROC FAR
        BCLR T3R                ; stop the timer
        BSET ADST               ; start an A/D conversion
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
LOCATE
locatein.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES(’RAM’ (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO locate.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;*********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
```

```
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
```

```
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     ‘CC0IO’
EX1IN           LIT     ‘CC1IO’
EX2IN           LIT     ‘CC2IO’
EX3IN           LIT     ‘CC3IO’

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
```

```
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     ’AN10’
T5EUD           LIT     ’AN11’
T6IN            LIT     ’AN12’
T5IN            LIT     ’AN13’
T4EUD           LIT     ’AN14’
T2EUD           LIT     ’AN15’

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22           DEFB    CCM5.11
ACC23           DEFB    CCM5.15

ACC24           DEFB    CCM6.3
ACC25           DEFB    CCM6.7
ACC26           DEFB    CCM6.11
ACC27           DEFB    CCM6.15

ACC28           DEFB    CCM7.3
ACC29           DEFB    CCM7.7
ACC30           DEFB    CCM7.11
ACC31           DEFB    CCM7.15

T2R             DEFB    T2CON.6
T2UD            DEFB    T2CON.7
T2UDE           DEFB    T2CON.8

T3R             DEFB    T3CON.6
T3UD            DEFB    T3CON.7
T3UDE           DEFB    T3CON.8
T3OE            DEFB    T3CON.9
T3OTL           DEFB    T3CON.10

T4R             DEFB    T4CON.6
T4UD            DEFB    T4CON.7
T4UDE           DEFB    T4CON.8

T5R             DEFB    T5CON.6
T5UD            DEFB    T5CON.7
T5UDE           DEFB    T5CON.8
T5CLR           DEFB    T5CON.14
T5SC            DEFB    T5CON.15

T6R             DEFB    T6CON.6
T6UD            DEFB    T6CON.7
T6UDE           DEFB    T6CON.8
T6OE            DEFB    T6CON.9
T6OTL           DEFB    T6CON.10
T6SR            DEFB    T6CON.15

T2IE            DEFB    T2IC.6
T2IR            DEFB    T2IC.7
T3IE            DEFB    T3IC.6
T3IR            DEFB    T3IC.7
T4IE            DEFB    T4IC.6
T4IR            DEFB    T4IC.7
T5IE            DEFB    T5IC.6
T5IR            DEFB    T5IC.7
T6IE            DEFB    T6IC.6
T6IR            DEFB    T6IC.7

CRIE            DEFB    CRIC.6
CRIR            DEFB    CRIC.7

S0TIE           DEFB    S0TIC.6
S0TIR           DEFB    S0TIC.7
S0RIE           DEFB    S0RIC.6
S0RIR           DEFB    S0RIC.7
S0EIE           DEFB    S0EIC.6
S0EIR           DEFB    S0EIC.7
S0TBIE          DEFB    S0TBIC.6
S0TBIR          DEFB    S0TBIC.7

SSCTIE          DEFB    SSCTIC.6
SSCTIR          DEFB    SSCTIC.7
```

```
SSCRIE          DEFB    SSCRIC.6
SSCRIR          DEFB    SSCRIC.7
SSCEIE          DEFB    SSCEIC.6
SSCEIR          DEFB    SSCEIC.7
SSCTE           LIT     'SSCTEN'
SSCRE           LIT     'SSCREN'
SSCPE           LIT     'SSCPEN'
SSCBE           LIT     'SSCBEN'


CC0IE           DEFB    CC0IC.6
CC0IR           DEFB    CC0IC.7
CC1IE           DEFB    CC1IC.6
CC1IR           DEFB    CC1IC.7
CC2IE           DEFB    CC2IC.6
CC2IR           DEFB    CC2IC.7
CC3IE           DEFB    CC3IC.6
CC3IR           DEFB    CC3IC.7
CC4IE           DEFB    CC4IC.6
CC4IR           DEFB    CC4IC.7
CC5IE           DEFB    CC5IC.6
CC5IR           DEFB    CC5IC.7
CC6IE           DEFB    CC6IC.6
CC6IR           DEFB    CC6IC.7
CC7IE           DEFB    CC7IC.6
CC7IR           DEFB    CC7IC.7
CC8IE           DEFB    CC8IC.6
CC8IR           DEFB    CC8IC.7
CC9IE           DEFB    CC9IC.6
CC9IR           DEFB    CC9IC.7
CC10IE          DEFB    CC10IC.6
CC10IR          DEFB    CC10IC.7
CC11IE          DEFB    CC11IC.6
CC11IR          DEFB    CC11IC.7
CC12IE          DEFB    CC12IC.6
CC12IR          DEFB    CC12IC.7
CC13IE          DEFB    CC13IC.6
CC13IR          DEFB    CC13IC.7
CC14IE          DEFB    CC14IC.6
CC14IR          DEFB    CC14IC.7
CC15IE          DEFB    CC15IC.6
CC15IR          DEFB    CC15IC.7
CC16IE          DEFB    CC16IC.6
CC16IR          DEFB    CC16IC.7
CC17IE          DEFB    CC17IC.6
CC17IR          DEFB    CC17IC.7
CC18IE          DEFB    CC18IC.6
CC18IR          DEFB    CC18IC.7
CC19IE          DEFB    CC19IC.6
CC19IR          DEFB    CC19IC.7
CC20IE          DEFB    CC20IC.6
CC20IR          DEFB    CC20IC.7
CC21IE          DEFB    CC21IC.6
CC21IR          DEFB    CC21IC.7
CC22IE          DEFB    CC22IC.6
CC22IR          DEFB    CC22IC.7
CC23IE          DEFB    CC23IC.6
CC23IR          DEFB    CC23IC.7
CC24IE          DEFB    CC24IC.6
CC24IR          DEFB    CC24IC.7
CC25IE          DEFB    CC25IC.6
CC25IR          DEFB    CC25IC.7
CC26IE          DEFB    CC26IC.6
CC26IR          DEFB    CC26IC.7
CC27IE          DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.8 CAN Router

On the next page starts the code for the CAN Router. The files for the node are as follows.

1. comp.bat
2. main.asm
3. serialApril.asm
4. cnmod.asm
5. canmo.asm
6. canint.asm
7. timers.asm
8. linker.lnv
9. Reg167b.def

comp.bat

```
a166 main.asm
a166 serialApril.asm
a166 timers.asm
a166 canmod.asm
a166 canmo.asm
a166 canint.asm
l166 LINK main.obj timers.obj serialApril.obj canint.obj canmod.obj canmo.obj TO main.ln
o
l166 @linker.lnv
ihex166 -i16 main.out -o main.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 512 Words

ASSUME DPP3:SYSTEM

EXTERN serial_init:FAR
EXTERN canin:FAR                ; Can function
EXTERN serial_timer_initialize:FAR; serial


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section

                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization


        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        MOV DP2, ONES
        NOP
        MOV P2, ZEROS
          ;; Initialize the Serial Port
              CALL serial_init
        ;; End of Serial Port Initialization
        ;; Initialize the serial port timer
                CALL serial_timer_initialize; pain in the ass
        ;; Initialize CAN Bus
              CALL canin        ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE          ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H       ; reset interrupt number is zero at 0h
        ORG 000H                  ; forces next instruction to be located at 0h
        JMP start                 ; installs a pointer to the startup routine
        RETI                      ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME serial_functions           ; Every File needs one of these
        ;; Declare a common register bank
        ;; This register bank is common to all files
        ;; which declare that they are going to use it.
RBANK1 COMREG R0-R15
        ;;Declare 'serial_init' global so other files can call it.
GLOBAL serial_init
GLOBAL byte_counter
GLOBAL confirm_message
GLOBAL message_transmitting
GLOBAL message_to_transmit


EXTERN CAN_message_BYTES:BYTE

        ;; GLOBAL serial_transmit_in_use
        ;; GLOBAL serial_transmit_requested
        ;; Assign the DPPs with the assume directive
        ;; this really doesn't do anything worth mentioning
        ;; nothing that I understand anyhow.
ASSUME DPP0:incoming_message, DPP1:transmit_structure, DPP3:SYSTEM


        ;; Declare the Data sections to be used by the
        ;; serial port.
incoming_message SECTION DATA BYTE GLOBAL 'RAM'
start_of_received_message label BYTE    ; For Looping later
        start_of_frame DSB 1
        number_of_bytes DSB 1   ; length of CAN message
        direction_of_transmission DSB 1
        message_id DSB 2
        message_data DSB 8
        check_sum DSB 2
        end_of_frame DSB 1      ; j3ll
        byte_counter DSW 1
incoming_message ENDS


transmit_structure SECTION DATA BYTE GLOBAL 'RAM'
        transmit_data DSB 16
        receive_buffer DSB 16
        transmit_counter DSW 1
        message_to_transmit DSW 1
        message_transmitting DSW 1
transmit_structure ENDS

serial_constants SECTION DATA BYTE GLOBAL 'ROM'
        resend_message DB '&!!Send  Over!!&'
        time_out_message DB '&!!Time   Out!!&'
        message_length DB 16
        data_structure_size DB 12
serial_constants ENDS

        ;; Start of the serial section code.  There are X functions in
        ;; 3 different sections this file.
        ;; In the 'serial_start' section there is
```

```
        ;; 'rechandler', 'receive_message'
serial_start SECTION CODE
serial_init PROC FAR
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
;; Initialize the Serial Port
        MOV DPP0, #PAG incoming_message
        MOV DPP1, #PAG transmit_structure
        AND DPP0:byte_counter, ZEROS    ; hjhjh
        AND DPP1:transmit_counter, ZEROS; jasdf
        AND DPP1:message_to_transmit, ZEROS; Clear the message to transmit
        AND DPP1:message_transmitting, ZEROS; CLEAR MESSAGE_TRANSMITTING
        MOV S0CON, #08011h  ;Sets the serial port
        MOV S0BG, #0040h    ;Sets the baud rate to 9600
        MOVB S0RIC, #030h  ;Sets the interrupt for the receive side
        MOV S0TBUF, ZEROS
        EXTR #1                     ; enables access to ESFR for 1 command only
        MOVB S0TBIC, #020h   ;Sets the interrupt handler for send buffer
        BSET S0RIC.6        ;enable the receive interrupt handler
        EXTR #1                     ; Enables access to ESFR
        BCLR S0TBIC.6        ;enable the send buffer interrupt handler
        MOV DP3, ONES       ;set the port direction to output
        MOV P3, ONES        ;set the outputs to 1

        BCLR DP3.11         ;Set the pin direction to input
        BCLR P3.11          ;Not a clue
;;  End of serial port initialization
        POP DPP2
        POP DPP1
        POP DPP0
        RET
serial_init ENDP
serial_start ENDS

serial_receive SECTION CODE
receive_handler PROC TASK INTNO=02BH
        ORG 0ACh
        CALL rechandler
        RETI
receive_handler ENDP

rechandler PROC FAR
        ;; The first part of this procedure makes sure that
        ;; the byte_counter which is the offset from the start
        ;; of the data array which is used to hold the data message is
        ;; set to the correct value
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH DPP0
        PUSH DPP1

        MOV DPP0, #PAG start_of_received_message
        MOV DPP1, #PAG message_length
        MOV R0, #DPP0:start_of_received_message;  me

        BCLR T5CON.6             ; start the timer
        MOV T5, #0001h           ; set the timer to 1
        MOV R2, DPP0:byte_counter
        ADD R0, R2       ;  me i
        MOVB [R0] , S0RBUF

        ADDB RL2, #01h
        MOV DPP0:byte_counter, R2
        ;; The structure is 14 bytes long so the comparison is
        ;; done against #0Ch.
        CMPB RL2, DPP1:message_length ;know when to call the handling function
        JMPA cc_Z, handle_message ; need to decode the message
        BSET T5CON.6            ; jkj
        JMP receive_end         ; exit function

handle_message:
        BCLR T5CON.6            ; TURN OFF THE TIMER
        MOV T5, #001h
        MOV DPP0:byte_counter, ZEROS

        CALL receive_message;j


receive_end:
        POP DPP1
        POP DPP0
        POP R2
        POP R1
        POP R0
        RET
rechandler ENDP


receive_message PROC FAR
        PUSH DPP0
        MOV DPP0, #PAG transmit_structure
        CALL test_checksum      ; necessary
        CALL do_the_CAN_JAZZ    ; setup and execute the CAN Message Object
;       CALL remove_from_receive_buffer; jkj
;       CMP ZEROS, DPP0:message_transmitting; jkj
;       JMP cc_NZ, exit_receive_message; jkj
;       CALL confirm_message    ; Necessary

exit_receive_message:

        POP DPP0
        RET
receive_message ENDP

remove_from_receive_buffer PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        MOV DPP0, #PAG start_of_received_message
        MOV DPP1, #PAG transmit_structure
        MOV DPP2, #PAG serial_constants
        AND R2, ZEROS
        MOV R0, #DPP0:start_of_received_message
        MOV R1, #DPP1:receive_buffer

move_received_data:
        MOVB [R1],[R0]
        ADD R2, #01h
        ADD R0, #01h
        ADD R1, #01h
        CMPB RL2, DPP2:message_length
        JMP cc_NZ, move_received_data
        POP DPP2
        POP DPP1
        POP DPP0
```

```
        POP R2
        POP R1
        POP R0
        RET
remove_from_receive_buffer ENDP
serial_receive ENDS

checksum_test_functions SECTION CODE
test_checksum PROC FAR
        PUSH R0                 ; To be used as a pointer to the message
        PUSH R1                 ; To be used as an accumulator
        PUSH R2                 ; To be used to contain data structure size
        PUSH R3                 ; To be used as a counter
        PUSH R4                 ; To be used for byte to word conversions
        PUSH R5
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        MOV DPP0,  #PAG start_of_received_message;  DPP0= message_id's page
        MOV DPP1, #PAG data_structure_size
        MOV DPP2, #PAG transmit_structure
        AND R1, ZEROS           ; Make the accumulator value = Zero
        AND R3, ZEROS           ; Set the loop counter to zero
        AND R4, ZEROS           ; Make R4 all zeros
        MOV R0, #DPP0:number_of_bytes; beginning of important data

calculate_total:                ; Loop through the entire data structure
        MOVB RL4, [R0+]
        ADD R1, R4              ; Byte to word conversion done here
        ADDB RL3, #01h          ; increment the loop count
        CMPB RL3, DPP1:data_structure_size ; Cmp R3 to the size of the loop
        JMP cc_NZ, calculate_total; If not equal then add again

        MOVB RH2, DPP0:check_sum
        MOVB RL2, DPP0:check_sum + 1
        CMP R1, R2        ;computed vs received checksums
        JMP cc_NZ, checksum_error
        MOV R5, #01h
        ADD DPP2:message_to_transmit, R5; Indicates good reply
        JMP exit_checksum

checksum_error:
        MOV R0, #02h            ; indicates checksum error
        ADD DPP2:message_to_transmit, R0

exit_checksum:
        POP DPP2
        POP DPP1
        POP DPP0
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
test_checksum ENDP
checksum_test_functions ENDS


serial_transmit SECTION CODE
confirm_message PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        ;; First thing to do is copy all data into the transmit
        ;; data data-structure
        ;; load DPP0 and DPP1 with the data pages of the two data structures
        MOV DPP0, #PAG start_of_received_message; old version
        MOV DPP1, #PAG transmit_structure
        MOV DPP2, #PAG serial_constants
        MOV R3, #01h
        NOP                     ; Random NOP
        MOV DPP1:message_transmitting, R3
        ;; determine which message to transmit

        NOP                     ; Another RANDOM NOP
        MOV R3, DPP1:message_to_transmit ; Move into R3 the message to transmit
        MOV R4, R3              ; Copy for fast recovery
        JMP setup_pointers      ; Test code
        ;; move the start addresses of the two data structures
        ;; into registers which are to be used as pointers to
        ;; the data structures
        AND R3, #01h            ; Isolate possible good message
        CMP R3, #01h            ; See if good message
        JMP cc_NZ, next_possibility1
        MOV R0, #DPP1:receive_buffer
        SUB R3, #01h
        MOV DPP1:message_to_transmit, R3
        JMP setup_pointers

next_possibility1:
        MOV R3, R4              ; Refresh R3 buffer
        AND R3, #02h            ; Isolate Possible Send Over
        CMP R3, #02h            ; See if Send Over exists
        JMP cc_NZ, next_possibility2
;       JMP exit_quickly        ; test only
;       MOV R0, #DPP1:receive_buffer; test code
        MOV R0, #DPP2:resend_message; jkj
        SUB R3, #02h
        MOV DPP1:message_to_transmit, R3

        JMP setup_pointers

next_possibility2:
        MOV R3, R4
        AND R3, #04h
        CMP R3, #04h
        JMP cc_NZ, next_possibility3
;       JMP exit_quickly        ; test only
        MOV R0, #DPP2:time_out_message; actual possibility
;       MOV R0, #DPP1:receive_buffer; test code
        SUB R3, #04h
        MOV DPP1:message_to_transmit, R3
        JMP setup_pointers

next_possibility3:
        MOV R3, R4
        AND R3, #08h
        CMP R3, #08h
        JMP cc_NZ, next_possibility4
;               MOV R0, #DPP1:receive_buffer; Test Code
        MOV DPP2, #PAG CAN_message_BYTES; actual possibility
        NOP
        MOV R0, #DPP2:CAN_message_BYTES; set R0 to point to address of CAN return me
ssage
        SUB R3, #08h
        MOV DPP1:message_to_transmit, R3
        JMP setup_pointers

next_possibility4:
        MOV DPP1:message_to_transmit, ZEROS
        ;MOV R0, #DPP1:receive_buffer; jkj
        MOV DPP2, #PAG CAN_message_BYTES; actual possibility
        NOP
        MOV R0, #DPP2:CAN_message_BYTES; set R0 to point to address of CAN return me
ssage

setup_pointers:
;       MOV R0, #DPP0:start_of_received_message; test purposes
;       MOV R0, #DPP1:receive_buffer; test code
        MOV DPP2, #PAG CAN_message_BYTES; test code
        NOP                     ; test code
        MOV R0, #DPP2:CAN_message_BYTES; test code
        MOV R1, #DPP1:transmit_data
        AND R2, ZEROS           ; set the counter to zero
        MOV DPP1:message_to_transmit, ZEROS

move_data:
        MOVB [R1], [R0] ; move data from message buffer to transmit buffer
        ADD R2, #01h            ; Increment everyone by #01h
        ADD R0, #01h
        ADD R1, #01h
        CMPB RL2, DPP2:message_length ;Check all  data has been transfered
        JMP cc_NZ, move_data    ; if more data to transfer then loop
        ;; The EXTR #1 instruction allows the BSET instruction
        ;; to access the Extended Special Function Register area.
        ;; without the EXTR #1 instruction, there is no way you can
        ;; access the S0TBIC register.  You also need the $EXTSFR and
        ;; the $EXTINSTR assembler controls (located at the top of
        ;; the file) for this to work.
        EXTR #1
        BSET S0TBIC.6

        MOV DPP1:transmit_counter, ZEROS

        ;; Calling a TRAP is a software way of creating an interrupt
        ;; in this case we are causing the interrupt handler for the
        ;; serial transmit buffer to occur.  The difference between
        ;; calling a trap and having the interrupt be generated from
        ;; a hardware event is that when calling a trap, the CPU
        ;; does not change priority level
        TRAP #047h              ; asdf
;       CALL transmit_buffer_function; Test Code

exit_quickly:
        POP DPP2
        POP DPP1
        POP DPP0
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
confirm_message ENDP

transmit_handler PROC TASK INTNO=047h
```

```
        ;; This is the interrupt handler for the Serial Transmit Buffer
        ;; Interrupt.  It is activated when data is transmitted from
        ;; the transmit buffer to the transmit shift register.
        ORG 011Ch
        CALL transmit_buffer_function
        RETI
transmit_handler ENDP

transmit_buffer_function PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH DPP1
        PUSH DPP2
        ;; make data page on have the page number for transmit_data
        MOV DPP1, #PAG transmit_data
        MOV DPP2, #PAG message_length

        ;; the following is curious.  It moves the address of transmit_data
        ;; into R0, but R0 is 16 bits and the address of transmit_data is
        ;; actually 24 bits...must be some assembler magic going on in the
        ;; background
        MOV R0, #DPP1:transmit_data
        NOP
        MOV R1, DPP1:transmit_counter; move the transmit_counter into R1
        MOVB RL2, DPP2:message_length   ; Go through the loop 12 times
        ;; The below add makes the value in R0 point to what ever it was
        ;; pointing to plus an offset which is in R1
        ADD R0, R1; increment the data pointer
        NOP
        ;; The problem that I encountered was that I was trying to
        ;; do a MOV from memory but the data type that was in memory
        ;; was a BYTE so the computer screwed up.

        MOVB S0TBUF, [R0]

        NOP
        ADDB RL1, #01h          ; Increment the transmit counter register
        MOVB DPP1:transmit_counter, RL1; move the value into the transmit counter

        CMPB RL1, DPP2:message_length   ; comp current count with final count
        JMP cc_NZ, exit_routine ; if they are equal then stop sending data


end_handler:
        EXTR #1                 ; necessary to access an Extended SFR
        BCLR S0TBIC.6           ; for some reason this register is an E-SFR
;       BSET S0RIC.6            ; asfd
;       BSET T5IE               ; asfdasd
;       EXTR #1                 ; kjkj
;       BSET XP0IC.6            ; asdfasd
        MOV DPP1:transmit_counter, ZEROS; reset the counter register
        AND DPP1:message_transmitting, ZEROS; Wait until all queued messages have transm
itted to clear
;       CMP ZEROS, DPP1:message_to_transmit; see if any more messages are waiting to tra
nsmit
;       JMP cc_Z, exit_routine  ; jkj
;       CALL confirm_message    ; jkj

exit_routine:

        POP DPP2
        POP DPP1                ; Pop all data off the stack
        POP R3
```

```
        POP R2
        POP R1
        POP R0
        RET
transmit_buffer_function ENDP

do_the_CAN_JAZZ PROC FAR
        PUSH R0
        PUSH R5
        PUSH R6
        PUSH DPP0
        NOP
        MOV DPP0, #PAG direction_of_transmission
        NOP
        MOV RL0, DPP0:direction_of_transmission
        CMPB RL0, #08h          ; See if it is a transmit frame
        JMP cc_Z, transmit_information; jump
        CMPB RL0, #0h           ; See if it is a remote frame
        JMP cc_UC, receive_information
        JMP exit_CAN_function

receive_information:
        MOV R5, #05555h         ; This code makes a message object valid
        MOV MCR_M2, R5          ; Now Message_object 2 is invalid and can be operate
d on

        ;; Set the message mask
        MOVB RH5, DPP0:message_id; jkjk
        MOVB RL5, DPP0:message_id + 1; jadsf
        MOV R6, #0EF22h
        NOP
        MOV [R6],R5

        ;; Generate the Message Configuration Register
        AND R6, ZEROS
        AND R5, ZEROS
        MOVB RL5, DPP0:direction_of_transmission
        MOVB RL6, DPP0:number_of_bytes
        SHL R6, #04h

        ADD R5, R6

        MOV MCD_M2, R5

        ;; put data into data register
        MOV R5, #DPP0:message_data
        ADD R5, #01h
        MOV RH6, [R5]
        ADD R5, #01h
        MOV RL6, [R5]
        MOV DATA_M2, R6
        ;; Now reactivate the Message Control Object
        MOV R5, #06599h         ; Valid, requested transmission, receive interrupt e
nabled
        MOV MCR_M2, R5
        ;MOV P2, #05555h                ; test pattern
        JMP exit_CAN_function

transmit_information:
        ;; Valid Messages get Sent to the CAN BUS
        ;; First The Message OBJECT Must be setup.
        ;; Message Object 1 is always used right now
        ;; First make the message invalid
        MOV R5, #05955h
        MOV MCR_M1, R5
        ;; Set the message mask
        MOVB RH5, DPP0:message_id
        MOVB RL5, DPP0:message_id + 1
        MOV R6, #0EF12h
        NOP
        MOV [R6],R5

        ;; Generate the Message Configuration Register
        AND R6, ZEROS
        AND R5, ZEROS
        MOVB RL5, DPP0:direction_of_transmission
        MOVB RL6, DPP0:number_of_bytes
        SHL R6, #04h

        ADD R5, R6
;       MOV P2, R5              ; Test code
        MOV MCD_M1, R5

        ;; put data into data register
        MOV R5, #DPP0:message_data
        ADD R5, #01h
        MOV RH6, [R5]
        ADD R5, #01h
        MOV RL6, [R5]
        MOV DATA_M1, R6
        ;; Now reactivate the Message Control Object
        MOV R5, #06595h
        MOV MCR_M1, R5

exit_CAN_function:

        POP DPP0
        POP R6
        POP R5
        POP R0
        RET
do_the_CAN_JAZZ ENDP
serial_transmit ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; E0FF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0071h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
        RET
canin   ENDP

setall PROC FAR                ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15             ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1          ; start of Message Object 1
        AND R1, ZEROS
        OR R1, #5555h            ; This MO is inactive and will be controlled from the PC
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R3 to
        OR R3, #0003h            ; message id for message object 1
        MOV [R2],R3              ; message id = #0003h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; put 0AAh into first data byte and set to transmit
        MOV MCD_M1,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M1, ZEROS       ; fill the Data of the MO with Zeros

        ;; set up second message object to be used with receive objects
        MOV R2, #MCR_M2          ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #05555h           ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO2's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R3 to
        OR R3, #0003h            ; message id for message object 2
        MOV [R2],R3              ; message id = #0003h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h            ; This guy is a receive object
        MOV MCD_M2,R1              ; Databyte(0) = 0 and Set to receive and 3 bytes of da
ta
        MOV DATA_M2, ZEROS       ; fill the Data of the MO with Zeros

        POP R3
        POP R2
        POP R1
```

```
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15              ; declare bank of 16 global registers

ASSUME DPP3:SYSTEM

EXTERN message_transmitting:WORD; from serialFebruary
EXTERN message_to_transmit:WORD
EXTERN confirm_message:FAR

GLOBAL CAN_message_BYTES
can_interrupt_data SECTION DATA WORD GLOBAL 'RAM'
        CAN_message_BYTES LABEL BYTE
        CAN_message_word_1 DSW 1
        CAN_message_word_2 DSW 1
        CAN_message_word_3 DSW 1
        CAN_message_word_4 DSW 1
        CAN_message_word_5 DSW 1
        CAN_message_word_6 DSW 1
        CAN_message_word_7 DSW 1
        CAN_message_word_8 DSW 1
can_interrupt_data ENDS

can_interrupts  SECTION CODE
can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        PUSH R8
        PUSH R9
        PUSH R10
        PUSH R11
        PUSH R12
        PUSH R13
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        MOV DPP0, #PAG CAN_interrupt_data

        MOV R0, #05555h         ; deactive code
        MOV MCR_M2, R0          ; Deactive the Second Message Object
        AND R7, ZEROS
        MOV R11, MCD_M2         ; Moves DLC and DIR into Lower Byte and DATA byte 0 into
 upper byte
        MOV R12, DATA_M2                ; Moves DATA byte 1 into RL2 and DATA byte 2 int
o RH2
        MOV R13, MID_M2         ; Moves the Message ID into Register 8
```

```
        MOV P2, R12                     ; jkasdjfjfkdls

        ;; Start building the message for serial transmission
        MOV R1, R11
        AND R1, #0F0h                   ; Isolate Data Length Code
        SHL R1, #04h                    ; Position it in RH1
        MOVB RL1, #0A0h                 ; Move message start bit into place

        ;; Isolate into the top part of the word the Direction of transmission
        MOV R2, R11                     ; Copy into R1
        AND R2, #08h                    ; Isolate the Direction of the data

        MOV R3, R13                     ; Start breaking down the message ID
        MOVB RH2, RH3                   ; Finish Word 2

        ;MOVB RH3, RL3                  ; Start Word 3
        MOVB RH3, #00h                  ; The First Byte of Data is Always ZERO so Move ZERO
S into RH3

        MOV R4, R12                     ; Start Word 4
        ;; Words 5 and 6 are just ZERO therefore don't use  a  register
        PUSH R3
        MOVB RH3, RH4
        MOVB RH4, RL4
        MOVB RL4, RH3
        POP R3

        ;; Now compute the Checksum
        AND R0, ZEROS
        AND R9, ZEROS
        ;; Don't user RH1 in the computation of the Checksum
        MOVB RL0, RH1                   ; BYTe to word conversion
        ADD R9, R0                      ; add the Data Length Code to the Checksum

        AND R0, ZEROS                   ; Reset the byte to word conversion buffer
        MOVB RL0, RH2
        ADD R9, R0                      ; add the Direction of transmission to Checksum

        AND R0, ZEROS
        MOVB RL0, RL2
        ADD R9, R0                      ; add the upper byte of the message id to the checks
um

        AND R0, ZEROS
        MOVB RL0, RH3                   ; add the lower byte of the message id to the checks
um
        ADD R9, R0

        AND R0, ZEROS
        MOVB RL0, RL3                   ; add the lower byte of the message id to the checks
um
        ADD R9, R0

        AND R0, ZEROS
        MOVB RL0, RH4                   ;jk
        ADD R9, R0                      ; add the upper byte of the message data to checksum

        AND R0, ZEROS
        MOVB RL0, RL4
        ADD R9, R0                      ; add lower byte of the data to checksum

        AND R0, ZEROS
        MOV R6, R9                      ; Move the checksum into a byte addressable register
        AND R5, ZEROS
        MOV RH5, RH6                    ; Move the upper byte of the checksum into R5
```

```
;       MOV  RL6                 ; test
        MOV RH6, #0Ah
        ;; THE CHECKSUM IS NOW COMPUTED


        ;; THE CAN MESSAGE IS NOW COMPLETED IN REGISTERS R1 THROUGH R8
        ;; Now put the CAN message into memory

        MOV DPP0:CAN_message_word_1, R1 ; put data into memory
        MOV DPP0:CAN_message_word_2, R2 ; put data into memory
        MOV DPP0:CAN_message_word_3, R3 ; put data into memory
        MOV DPP0:CAN_message_word_4, R4 ; put data into memory
        MOV DPP0:CAN_message_word_5, ZEROS ; put data into memory
        MOV DPP0:CAN_message_word_6, ZEROS ; put data into memory
        MOV DPP0:CAN_message_word_7, R5 ; put data into memory
        MOV DPP0:CAN_message_word_8, R6 ; put data into memory

        MOV R0, #05599h
        MOV MCR_M2, R0           ; Reactive second Message Object

        MOV DPP1, #PAG message_transmitting
        MOV DPP2, #PAG message_to_transmit
        MOV R0, #08h
        ADD DPP2:message_to_transmit, R0
        CMP ZEROS, DPP1:message_transmitting; test
        JMP cc_Z, CAN_to_transmit; test
        JMP cc_UC, exit_can      ; test

CAN_to_transmit:
        MOV R0, DPP2:message_to_transmit
        PUSH R0
        MOV R1, #08h
        MOV DPP2:message_to_transmit, R1
        CALL confirm_message     ; test
        POP R0
        MOV DPP2:message_to_transmit, R0

exit_can:
        POP DPP2
        POP DPP1
        POP DPP0
        POP R13
        POP R12
        POP R11
        POP R10
        POP R9
        POP R8
        POP R7
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
can_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
$SEGMENTED                          ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                            ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15
GLOBAL serial_timer_initialize

EXTERN confirm_message:FAR      ; create pointer to time_out_error
EXTERN byte_counter:WORD; Get Reference to byte_counter
EXTERN message_transmitting:WORD; message_transmitting is a global variable

EXTERN message_to_transmit:WORD

serial_timer SECTION CODE
serial_timer_initialize PROC FAR
        MOV T5CON, #0000h           ; setup GPT2 Auxiliary Timer T5
        ;; had a problem with the level of the timer interrupt
        ;; with respect to that of the serial receive interrupt
        ;; needed to make the timer interrupt higher than that of
        ;; the serial receive interrupt.
        MOV T5IC, #002Bh
        MOV T5, #0001h
        BSET T5IE
        RET
serial_timer_initialize ENDP

serial_timer_interrupt PROC TASK INTNO=025H
        ORG 094H
        CALL serial_timer_handler; the timer handler
        RETI
serial_timer_interrupt ENDP

serial_timer_handler PROC FAR
        PUSH DPP0
        PUSH DPP1

        PUSH R0
        BCLR T5CON.6                ; turn off the timer
        MOV T5, #0001h              ; Reset the timer

        MOV DPP0, #PAG byte_counter
        MOV DPP1, #PAG message_transmitting
        MOV DPP0:byte_counter,ZEROS; Reset the receive buffer



error_reply:

        ADD R0, #04h                ; jaskjdf;
        MOV DPP1:message_to_transmit, R0; jkjkjk

        CMP ZEROS, DPP1:message_transmitting
        JMP cc_NZ, timer_return

        CALL confirm_message


timer_return:
```

```
        POP R0

        POP DPP1
        POP DPP0
        RET
serial_timer_handler ENDP
serial_timer ENDS
END



        MOV R0,#01h
        MOV DPP1:message_waiting_to_transmit, R0

        MOV R0, #02h
        MOV DPP1:waiting_message, R0
```

```
LOCATE
main.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES('RAM' (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO main.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;********************************************************************
C1CSR                   DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA  0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h
MID_M1          DEFA    0EF12h
MID_M2          DEFA    0EF22h
MID_M3          DEFA    0EF32h
MID_M4          DEFA    0EF42h
MID_M5          DEFA    0EF52h
MID_M6          DEFA    0EF62h
MID_M7          DEFA    0EF72h
```

```
MID_M8          DEFA    0EF82h
MID_M9          DEFA    0EF92h
MID_MA          DEFA    0EFA2h
MID_MB          DEFA    0EFB2h
MID_MC          DEFA    0EFC2h
MID_MD          DEFA    0EFD2h
MID_ME          DEFA    0EFE2h


DP8             DEFR    0FFD6h
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
```

```
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
```

```
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     ‘CC0IO’
EX1IN           LIT     ‘CC1IO’
EX2IN           LIT     ‘CC2IO’
EX3IN           LIT     ‘CC3IO’

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
```

```
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     ‘AN10’
T5EUD           LIT     ‘AN11’
T6IN            LIT     ‘AN12’
T5IN            LIT     ‘AN13’
T4EUD           LIT     ‘AN14’
T2EUD           LIT     ‘AN15’

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
```

```
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
ACC22           DEFB    CCM5.11
ACC23           DEFB    CCM5.15

ACC24           DEFB    CCM6.3
ACC25           DEFB    CCM6.7
ACC26           DEFB    CCM6.11
ACC27           DEFB    CCM6.15

ACC28           DEFB    CCM7.3
ACC29           DEFB    CCM7.7
ACC30           DEFB    CCM7.11
ACC31           DEFB    CCM7.15

T2R             DEFB    T2CON.6
T2UD            DEFB    T2CON.7
T2UDE           DEFB    T2CON.8

T3R             DEFB    T3CON.6
T3UD            DEFB    T3CON.7
T3UDE           DEFB    T3CON.8
T3OE            DEFB    T3CON.9
T3OTL           DEFB    T3CON.10

T4R             DEFB    T4CON.6
T4UD            DEFB    T4CON.7
T4UDE           DEFB    T4CON.8

T5R             DEFB    T5CON.6
T5UD            DEFB    T5CON.7
T5UDE           DEFB    T5CON.8
T5CLR           DEFB    T5CON.14
T5SC            DEFB    T5CON.15

T6R             DEFB    T6CON.6
T6UD            DEFB    T6CON.7
T6UDE           DEFB    T6CON.8
T6OE            DEFB    T6CON.9
T6OTL           DEFB    T6CON.10
T6SR            DEFB    T6CON.15

T2IE            DEFB    T2IC.6
T2IR            DEFB    T2IC.7
T3IE            DEFB    T3IC.6
T3IR            DEFB    T3IC.7
T4IE            DEFB    T4IC.6
T4IR            DEFB    T4IC.7
T5IE            DEFB    T5IC.6
T5IR            DEFB    T5IC.7
T6IE            DEFB    T6IC.6
T6IR            DEFB    T6IC.7

CRIE            DEFB    CRIC.6
CRIR            DEFB    CRIC.7

S0TIE           DEFB    S0TIC.6
```

```
S0TIR           DEFB    S0TIC.7
S0RIE           DEFB    S0RIC.6
S0RIR           DEFB    S0RIC.7
S0EIE           DEFB    S0EIC.6
S0EIR           DEFB    S0EIC.7
S0TBIE          DEFB    S0TBIC.6
S0TBIR          DEFB    S0TBIC.7

SSCTIE          DEFB    SSCTIC.6
SSCTIR          DEFB    SSCTIC.7
SSCRIE          DEFB    SSCRIC.6
SSCRIR          DEFB    SSCRIC.7
SSCEIE          DEFB    SSCEIC.6
SSCEIR          DEFB    SSCEIC.7
SSCTE           LIT     'SSCTEN'
SSCRE           LIT     'SSCREN'
SSCPE           LIT     'SSCPEN'
SSCBE           LIT     'SSCBEN'


CC0IE           DEFB    CC0IC.6
CC0IR           DEFB    CC0IC.7
CC1IE           DEFB    CC1IC.6
CC1IR           DEFB    CC1IC.7
CC2IE           DEFB    CC2IC.6
CC2IR           DEFB    CC2IC.7
CC3IE           DEFB    CC3IC.6
CC3IR           DEFB    CC3IC.7
CC4IE           DEFB    CC4IC.6
CC4IR           DEFB    CC4IC.7
CC5IE           DEFB    CC5IC.6
CC5IR           DEFB    CC5IC.7
CC6IE           DEFB    CC6IC.6
CC6IR           DEFB    CC6IC.7
CC7IE           DEFB    CC7IC.6
CC7IR           DEFB    CC7IC.7
CC8IE           DEFB    CC8IC.6
CC8IR           DEFB    CC8IC.7
CC9IE           DEFB    CC9IC.6
CC9IR           DEFB    CC9IC.7
CC10IE          DEFB    CC10IC.6
CC10IR          DEFB    CC10IC.7
CC11IE          DEFB    CC11IC.6
CC11IR          DEFB    CC11IC.7
CC12IE          DEFB    CC12IC.6
CC12IR          DEFB    CC12IC.7
CC13IE          DEFB    CC13IC.6
CC13IR          DEFB    CC13IC.7
CC14IE          DEFB    CC14IC.6
CC14IR          DEFB    CC14IC.7
CC15IE          DEFB    CC15IC.6
CC15IR          DEFB    CC15IC.7
CC16IE          DEFB    CC16IC.6
CC16IR          DEFB    CC16IC.7
CC17IE          DEFB    CC17IC.6
CC17IR          DEFB    CC17IC.7
CC18IE          DEFB    CC18IC.6
CC18IR          DEFB    CC18IC.7
CC19IE          DEFB    CC19IC.6
CC19IR          DEFB    CC19IC.7
CC20IE          DEFB    CC20IC.6
CC20IR          DEFB    CC20IC.7
CC21IE          DEFB    CC21IC.6
CC21IR          DEFB    CC21IC.7
CC22IE          DEFB    CC22IC.6
```

```
CC22IR          DEFB    CC22IC.7
CC23IE          DEFB    CC23IC.6
CC23IR          DEFB    CC23IC.7
CC24IE          DEFB    CC24IC.6
CC24IR          DEFB    CC24IC.7
CC25IE          DEFB    CC25IC.6
CC25IR          DEFB    CC25IC.7
CC26IE          DEFB    CC26IC.6
CC26IR          DEFB    CC26IC.7
CC27IE          DEFB    CC27IC.6
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15
```

```
SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15

PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.9 Data Acquisition Node

On the next page starts the code for the Data Acquisiton Node. The files for the node are as follows.

1. comp.bat
2. main.asm
3. cnmod.asm
4. canmo.asm
5. canint.asm
6. timers.asm
7. atod.asm
8. ema.asm
9. linker.lnv
10. Reg167b.def

B.10 DC/DC Converter Node

On the next page starts the code for the CAN Router. The files for the node are as follows.

1. comp.bat
2. main.asm
3. cnmod.asm
4. canmo.asm
5. canint.asm
6. linker.lnv
7. Reg167b.def

```
del *.obj
del *.lno
del *.out
del *.hex
a166 main.asm
a166 timers.asm
a166 atod.asm
a166 canmod.asm
a166 canmo.asm
a166 ema.asm
l166 LINK main.obj timers.obj atod.obj canmod.obj canmo.obj ema.obj TO main.lno
l166 @linker.lnv
ihex166 -i16 main.out -o main.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function
EXTERN atod_initialize:FAR              ; external atod initialization
EXTERN atod_timer_initialize:FAR


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Use Hysteresis for Special Input Thresholds
                EXTR #1
                BSET PICON.1
        ;; End of Setting Hysteresis for Special Input Thresholds

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h          ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization

        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
                BCLR DP2.0      ; Pins zero and on are used to capture the direction of
the current flow.
                BCLR DP2.1

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize the Analog to Digital Converter
                CALL atod_initialize; atod
        ;; End of A/D initialization

        ;; Initialize the timer for that controls A/D interval times
                CALL atod_timer_initialize
```

```
        ;; End of initialization for the timer that controls the A/D interval times

        ;; Initialize CAN Bus
              CALL canin       ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15           ; define a common register area of 16 registers
GLOBAL  canin                   ; The function must be declared Global at the
                                ; beginning of the module

EXTERN  canmocfg:FAR           ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE           ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; E0FF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        EXTR #2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        BCLR XP0IC.6    ; Turn off interrupts
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
        RET
canin   ENDP

setall PROC FAR                 ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15            ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
          PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system

        ;; This message object is the 36v battery voltage and should send the informatio
n if
        ;; it is requested by another node
        MOV R2, #MCR_M1         ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5555h           ; Make sure that this message object is invalid before o
perating on it
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to
        OR R3, #0005h           ; message id for message object 1
        MOV [R2],R3             ; message id = #0005h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; put 0AAh into first data byte and set to transmit
      MOV MCD_M1,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS      ; fill the Data of the MO with Zeros
        MOV R0, #05595          ; This makes a message object valid, but with no interru
pts
        MOV MCR_M1, R0          ; Message control Register 1 is now valid

        ;; This message object is the 36v battery current and direction information
        ;; it is set up to transmit the information if it is requested by another node
        MOV R2, #MCR_M2         ; start of Message Object 2
        AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1     ; set MO2's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R3 to
        OR R3, #0006h           ; message id for message object 2
        MOV [R2],R3             ; message id = #0006h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; This is a transmit object
        MOV MCD_M2,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M2, ZEROS      ; fill the Data of the MO with Zeros
        MOV R0, #05595          ; This makes a message object valid, but with no int
errupts
        MOV MCR_M2, R0          ; Message control Register 2 is now valid

;; This message object is the 36v battery temperature message object
        MOV R2, #MCR_M3         ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1                     ; set MO3's Control register to inactive

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to zero
        OR R3, #0007h           ; message id for message object 3
        MOV [R2],R3             ; message id = #0007h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h           ; This guy is a transmit object
        MOV MCD_M3,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes o
f data
        MOV DATA_M3, ZEROS      ; fill the Data of the MO with Zeros
        MOV R0, #05595          ; This makes a message object valid, but with no int
errupts
        MOV MCR_M3, R0          ; Message control Register 3 is now valid

;; This is the 36v battery state of charge message object
;; it is set up to transmit the state of charge at the request of another message ob
ject
;; it is different than the other message objects because it has a data length of 5
        MOV R2, #MCR_M4         ; start of Message Object 4
        AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1     ; set MO2's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
        AND R3, ZEROS             ; set R3 to
        OR R3, #0008h           ; message id for message object 4
        MOV [R2],R3             ; message id = #0009h
        ADD R2, #2h             ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS         ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0058h           ; This guy is a transmit object
      MOV MCD_M4,R1             ; Databyte(0) = 0 and Set to receive and 3 bytes of
data
        MOV DATA_M41, ZEROS     ; fill the Data of the MO with Zeros
        MOV DATA_M42, ZEROS     ; Clear this part of the message object too
        MOV R0, #05595          ; This makes a message object valid, but with no int
errupts
        MOV MCR_M4, R0          ; Message control Register 4 is now valid

;;  This is the 12v battery voltage message object
;;  It is a transmit message object with data length of 3
        MOV R2, #MCR_M5         ; start of Message Object 5
        AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1     ; set MO5's Control register

        ADD R2,#2h              ; point to Upper Arbitration register
      AND R3, ZEROS             ; set R3 to
        OR R3, #0009h           ; message id for message object 5
        MOV [R2],R3             ; message id = #0009h
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; This guy is a transmit object
     MOV MCD_M5,R1               ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M5, ZEROS       ; fill the Data of the MO with Zeros
        MOV R0, #05595           ; This makes a message object valid, but with no interru
pts
        MOV MCR_M5, R0           ; Message control Register 5 is now valid

;;  This is the 12v battery current and direction message object
;; it will transmit this information at the request of a remote from
        MOV R2, #MCR_M6          ; start of Message Object 6
     AND R1, ZEROS
        OR R1, #05555h           ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1      ; set MO6's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
     AND R3, ZEROS               ; set R3 to
        OR R3, #000BAh           ; message id for message object 6
        MOV [R2],R3              ; message id = #000Ah
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; This guy is a transmit object
     MOV MCD_M6,R1               ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M6, ZEROS       ; fill the Data of the MO with Zeros
        MOV R0, #05595           ; This makes a message object valid, but with no interru
pts
        MOV MCR_M6, R0           ; Message control Register 6 is now valid

;;  This is the 12v battery temperature message object
;;  It is setup to transmit the temperature information if an appropriate remote from is
 received
        MOV R2, #MCR_M7          ; start of Message Object 7
        AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1      ; set MO7's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
        AND R3, ZEROS              ; set R3 to
        OR R3, #000Bh            ; message id for message object 7
        MOV [R2],R3              ; message id = #000Bh
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; This is a transmit object with 3 data bytes
     MOV MCD_M7,R1               ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M7, ZEROS       ; fill the Data of the MO with Zeros
        MOV R0, #05595           ; This makes a message object valid, but with no interru
pts
        MOV MCR_M7, R0           ; Message control Register 7 is now valid

;; This message object contains the 12v battery state of charge.
;; It is similar to message object 4 in that it is setup to transmit 5 data bytes
        MOV R2, #MCR_M8          ; start of Message Object 8
     AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1      ; set MO8's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
     AND R3, ZEROS               ; set R3 to
        OR R3, #0000Ch           ; message id for message object 8
        MOV [R2],R3              ; message id = #000Ch
```

```
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; This guy is a transmit object
     MOV MCD_M8,R1               ; Databyte(0) = 0 and Set to receive and 3 bytes of
data
        MOV DATA_M81, ZEROS      ; fill the Data of the MO with Zeros
        MOV DATA_M82, ZEROS      ; fill the Data of the MO with Zeros
        MOV R0, #05595           ; This makes a message object valid, but with no int
errupts
        MOV MCR_M8, R0           ; Message control Register 8 is now valid

;; This message object is set up to transmit the state of the DC/DC converter
;; The state of the DC/DC converter is the output of the Energy Management algorithm
        MOV R2, #MCR_M9          ; start of Message Object 9
     AND R1, ZEROS
        OR R1, #05555h
        MOV [R2],R1              ; set MO2's Control register

        ADD R2,#2h               ; point to Upper Arbitration register
     AND R3, ZEROS               ; set R3 to
        OR R3, #0000Eh           ; message id for message object 8
        MOV [R2],R3              ; message id = #000Ch
        ADD R2, #2h              ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS          ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h            ; This guy is a transmit object
     MOV MCD_M9,R1               ; Databyte(0) = 0 and Set to receive and 3 bytes of
data
        MOV DATA_M9, ZEROS       ; fill the Data of the MO with Zeros
        MOV R0, #05595           ; This makes a message object valid, but with no int
errupts
        MOV MCR_M9, R0           ; Message control Register 9 is now valid

        POP R3
     POP R2
     POP R1
        POP R0
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED                      ; These are assembler controls
$EXTEND
$EXTSFR
$EXTMEM
$EXTINSTR
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS                        ; Assembler controls end here

NAME timer_functions
ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15
GLOBAL timer_period

GLOBAL atod_timer_initialize
atod_timer_data SECTION DATA WORD GLOBAL 'ROM'
        timer_period DW 04990h  ; This value plus the time necessary for all conversions
 is about 1 second
atod_timer_data ENDS

atod_timer SECTION CODE
atod_timer_initialize PROC FAR
        PUSH DPP0
        MOV DPP0, #PAG atod_timer_data
        MOV T3CON, #0086h       ; setup Core Timer T3 for count down mode
        MOV T3IC, #002Bh        ; Interrupt stuff
        BSET T3IE       ; enable the interrupt
        MOV T3, DPP0:timer_period       ; This value plus the time for all conversions i
s 1 second
        BSET T3CON.6
        POP DPP0
        RET
atod_timer_initialize ENDP


atod_interrupt PROC TASK INTNO=023h
        ORG 08Ch
        CALL atod_timer_handler
        RETI
atod_interrupt ENDP


atod_timer_handler PROC FAR
        PUSH DPP0
        PUSH R0
        MOV DPP0, #PAG atod_timer_data
        BCLR T3R                ; stop the timer
        MOV T3, DPP0:timer_period       ; Reset the count down register
        BSET ADST               ; start an A/D conversion
        POP R0
        POP DPP0
        RET
atod_timer_handler ENDP
atod_timer ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                             ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name atod

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15
EXTERN energy_management_algorithm:FAR
GLOBAL atod_initialize
GLOBAL voltage_36v
GLOBAL current_36v
GLOBAL current_direction_36v
GLOBAL temperature_36v
GLOBAL soc_36v_high_word
GLOBAL soc_36v_low_word
GLOBAL voltage_12v
GLOBAL current_12v
GLOBAL current_direction_12v
GLOBAL temperature_12v
GLOBAL soc_12v_high_word
GLOBAL soc_12v_low_word
GLOBAL soc_region_36v
GLOBAL soc_region_12v
GLOBAL r1_soc_36v_high
GLOBAL r1_soc_36v_low
GLOBAL r2_soc_36v_high
GLOBAL r2_soc_36v_low
GLOBAL r3_soc_36v_high
GLOBAL r3_soc_36v_low
GLOBAL r4_soc_36v_high
GLOBAL r4_soc_36v_low

GLOBAL r1_soc_12v_high
GLOBAL r1_soc_12v_low
GLOBAL r2_soc_12v_high
GLOBAL r2_soc_12v_low
GLOBAL r3_soc_12v_high
GLOBAL r3_soc_12v_low
GLOBAL r4_soc_12v_high
GLOBAL r4_soc_12v_low


        ;; This A/D is set up to measure the current in two different
        ;; loads.  Because this software is to be used as part of
        ;; 42volt bus node 1, it uses the names of the loads that
        ;; that node is supposed to control.
        ;; The analog to digital converter uses Port 5

atod_data_section SECTION DATA WORD GLOBAL 'RAM'
        voltage_36v DSW 1
        current_36v DSW 1
        current_direction_36v DSW 1
        temperature_36v DSW 1           ; Collected, but not used because no sensor is h
ooked up
        soc_36v_high_word DSW 1                         ; The 36v Battery STATE of charg
e
        soc_36v_low_word DSW 1
        soc_region_36v DSW 1    ; This is the SOC Region (1->5) in which the Battery is
operating

        voltage_12v DSW 1
        current_12v DSW 1
        current_direction_12v DSW 1
        temperature_12v DSW 1           ; collected, but not used because no sensor
is hooked up 5/5/99
        soc_12v_high_word DSW 1                         ; The 12v Battery STATE of c
harge
        soc_12v_low_word DSW 1
        soc_region_12v DSW 1    ; This is the SOC Region (1->5) in which the Battery
 is operating

        ;; These variables help with the computation
        total_period DSW 1
atod_data_section ENDS

battery_model_parameters SECTION DATA WORD GLOBAL 'ROM'
        starting_charge_36v_low DW      063E6h
        starting_charge_36v_high DW     010h
        starting_charge_12v_low DW      076A0h
        starting_charge_12v_high DW     025h
        r1_soc_36v_high DW  012h
        r1_soc_36v_low DW 07A0h
        r2_soc_36v_high DW 011h
        r2_soc_36v_low DW 035DCh
        r3_soc_36v_high DW 0Dh
        r3_soc_36v_low DW 0EE86h
        r4_soc_36v_high DW 09h
        r4_soc_36v_low DW 0D58Ah

        r1_soc_12v_high DW 029h
        r1_soc_12v_low DW 0359Ch
        r2_soc_12v_high DW 027h
        r2_soc_12v_low DW 05650h
        r3_soc_12v_high DW 01Fh
        r3_soc_12v_low DW 0D7F4h
        r4_soc_12v_high DW 016h
        r4_soc_12v_low DW 07A4Ch
battery_model_parameters ENDS

atod_setup SECTION CODE
atod_initialize PROC FAR
        ;; Initialize variables
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        ;; This section of code simply clears all of the variables which are to be u
sed during
        ;; data collection.
        ;; It also initializes the amphours of each of the batteries
        ;; The idea is that the system will boot up thinking that both of the batter
ies are okay
        ;; Then it will take and measure the voltages and determine from a graph whi
ch is figure xxx
        ;; in the master's thesis of James Geraci, what the actual state of charge i
s.
        MOV DPP0, #PAG atod_data_section
        MOV DPP1, #PAG battery_model_parameters
        AND DPP0:voltage_36v, ZEROS
        MOV R0, DPP1:starting_charge_36v_low
        AND DPP0:current_36v, ZEROS
        MOV R1, DPP1:starting_charge_36v_high

        AND DPP0:current_direction_36v, ZEROS
        AND DPP0:temperature_36v, ZEROS

        AND DPP0:soc_36v_high_word, R1
        AND DPP0:soc_36v_low_word, R0

        AND DPP0:voltage_12v, ZEROS
        AND DPP0:current_direction_12v, ZEROS

        MOV R0, DPP1:starting_charge_12v_low
        AND DPP0:temperature_12v, ZEROS

        MOV R1, DPP1:starting_charge_12v_high

        AND DPP0:soc_12v_high_word, R1
        AND DPP0:soc_12v_low_word, R0

        ;; Calculate the total conversion time to be used in calculating the amount of c
harge collected
        ;; Having a hard time understanding floating points in assembly so I'm just goin
g to make the total
        ;; period equal to 1

        ;; This below line of code setups up the A/D converter
        ;; for 6 channels and single conversion.
        ;; The idea is that the converter is on a timer
        ;; After each successful round of data collection, it will tell the
        ;; DC/DC converter that the data is ready, and the DC/DC converter
        ;; will then request transmission of each piece of information
        ;; It is also set for "Wait for read mode"
        ;; so the converter will wait for the user program to read
        ;; the buffer before processing the next channel.
        MOV ADCON, #0A225h      ; setup A/D control register

        ;; The below code sets up the A/D's Interrupt control register
        ;; The A/D is setup to have a group of 2 and a level of 10
        MOV ADCIC, #007Ah
        POP R3
        POP R2
        POP R1
        POP R0
        POP DPP2
        POP DPP1
        POP DPP0
        RET
atod_initialize ENDP
atod_setup ENDS

atod_handlers SECTION CODE
        atod_handler PROC TASK INTNO=028h
                   ORG 0A0H
                     CALL atod_function
                     RETI
        atod_handler ENDP

atod_function PROC FAR
        ;; this function works by seeing if the converter is converting
        ;; for the heater_measurement.  If the bit is set, then
        ;; the bit gets cleared and the IP jumps to where the
        ;; value in the converter is moved into the heater_current
        ;; variable.
```

```
        ;; otherwise the bit gets set and the value is moved into
        ;; the heated_rear_window_current variable

        ;; The Order of Conversion is:
        ;; 1) 36v temperature
        ;; 2) 12v temperature
        ;; 3) 36v voltage
        ;; 4) 12v voltage
        ;; 5) 12v current
        ;; 6) 36v current

        ;; The channels of the A/D are
        ;; 0) 36v current
        ;; 1) 12v current
        ;; 2) 12v voltage
        ;; 3) 36v voltage
        ;; 4) 12v temperature
        ;; 5) 36v temperature
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        PUSH R8
        PUSH R9
        MOV DPP0, #PAG atod_data_section
        MOV DPP1, #PAG battery_model_parameters

        MOV R0, ADDAT   ; Get the information from the A/D converter

        AND R1, ZEROS   ; Clear R1
        MOVB RL1, RH0           ; The upper nibble of the upper byte of the A/D info
rmation gives channel information
        SHR R1, #04h    ; Shift R1 right one nibble.  this puts the converter number
 into the lower nibble of R1

        MOV R7, R0              ; Make a copy of the current information

        ;; This piece of code isolates the DATA that has just been collected
        AND R0, #03FFh  ; This makes the upper 6 bytes zero

        ;; This code decides which piece of information has just been collected
        ;; and goes to the appropriate handler routine
        CMPB RL1, #05h  ; This tests to see if the conversion that just finished was
 made by converter number 5
        JMP cc_Z, temperature_36v_routine       ; Converter number 5 should take in
the temperature for the 36v battery
        NOP

        CMPB RL1, #04h  ; This tests to see if the conversion that just finished was
 made by converter number 4
        JMP cc_z, temperature_12v_routine       ; Converter number 4 should take in
the temperature for the 12v battery
        NOP

        CMPB RL1, #03h  ; This tests to see if the conversion that just finished was
 made by converter number 3
        JMP cc_z, voltage_36v_routine   ; Converter number 3 should take in the volt
age for the 36v battery
```

```
        NOP

        CMPB RL1, #02h  ; This tests to see if the conversion that just finished was mad
e by converter number 2
        JMP cc_z, voltage_12v_routine   ; Converter number 2 should take in the voltage
for the 12v battery
        NOP

        CMPB RL1, #01h  ; This tests to see if the conversion that just finished was mad
e by converter number 1
        JMP cc_z, current_12v_routine   ; Converter number 1 should take in the current
for the 12v battery
        NOP

        CMPB RL1, #00h  ; This tests to see if the conversion that just finished was mad
e by converter number 0
        JMP cc_z, current_36v_routine   ; Converter number 0 should take in the current
for the 36v battery
        NOP

temperature_36v_routine:
        ;; The information for the temp of 36v battery goes into CAN MO 3
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M3, R2          ; Turn Off the Message Control Register

        MOV DATA_M3, R0 ; Put the Data that has just been collected into Message Object
3
        MOV DPP0:temperature_36v, R0    ;put the data into memory
        MOV MCR_M3, R1
        JMP exit_routine

temperature_12v_routine:
        ;; The information for the temp of 12v battery goes into CAN MO 7
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M7, R2          ; Turn Off the Message Control Register

        MOV DATA_M7, R0 ; Put the Data that has just been collected into Message Object
7
        MOV DPP0:temperature_12v, R0    ; Put the 12v temperature into memory
        MOV MCR_M7, R1
        JMP exit_routine

voltage_36v_routine:
        ;; The information for the voltage of 36v battery goes into CAN MO 1
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M1, R2          ; Turn Off the Message Control Regis
        MOV DATA_M1, R0 ; Put the Data that has just been collected into Message Object
1
        MOV MCR_M1, R1
        JMP exit_routine

voltage_12v_routine:
        ;; The information for the voltage of 12v battery goes into CAN MO 5
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M5, R2          ; Turn Off the Message Control Register
        MOV DATA_M5, R0 ; Put the Data that has just been collected into Message Object
5
        MOV MCR_M5, R1
        JMP exit_routine

current_12v_routine:
        ;; The information for the current of the 12v battery goes into CAN MO 6
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M6, R2          ; Turn Off the Message Control Register
        MOV R8, #05595h         ; SAVE the configuration for MCR8 which is the 12v S
OC message object
        MOV MCR_M8, R2          ; Turn off MC8

        ;; The State of Charge of the Battery is also generated Here
                ;; The current measurement must be converted back into the actual cu
rrent value
                MOV R3, DPP0:soc_12v_low_word   ; The Low byte of the 12v battery so
c
                MOV R4, DPP0:soc_12v_high_word  ; The upper byte of the 12v battery
soc

                ;; Now we must check to see if the charge is positive or negative
                ;; This can be done for the 12v battery by checking to see if pin P2
.1 is a one or a zero
                MOV R2, P2
                AND R2, #0002h  ; This isolates the pin P2.1

                CMP R2, #0002h  ; This performs the comparison and sets the Z condit
ion flag
                JMP cc_NZ, perform_addition ;The Pin is brought Low when the Battery
 is charging
perform_subtraction:                    ; The battery is discharging
                SUB R3, R0
                SUBC R4, ZEROS
                JMP continue_data_collection
perform_addition:                               ; The battery is charging
                ADD R3, R0
                ADDC R4, ZEROS

                ;; When this point is reached the SOC should be in registers R3 and
R4.  The total charge for this period
                ;; should be in R0, the current direction should be in R2, and the c
urrent magnitude should be in R7
continue_data_collection:

        MOV DPP0:current_12v, R0        ; Put the current into memory
        MOV DPP0:current_direction_12v, R2 ; Put the current direction into memory
        MOV DPP0:soc_12v_high_word, R4  ; Put the upper part of the SOC into memory
        MOV DPP0:soc_12v_low_word, R3           ; Put the lower part of the SOC into
 memory
        MOVB RH2, RL2   ; Move the current direction into the upper byte of R2
        AND R2,  #00F00h        ; Get rid of all but the 3rd nibble
        SHL R2, #04h    ; Move the direction information into the upper nibble
        ADD R2, R0              ; Move the magnitude of the current into R2
        MOV DATA_M6, R2 ; Put the Data that has just been collected into Message Obj
ect 6

;; These lines put the SOC into the CAN message object number 8
        MOV DATA_M81, R4        ; Put the high data byte into data registers 2 and 1
        MOV DATA_M82, R3        ; Put the low data byte into data registers 4 and 3

        MOV MCR_M8, R8  ; Restore the SOC Message Object
        MOV MCR_M6, R1  ; Restore the CAN message object to operational status
        JMP exit_routine

current_36v_routine:
        ;; The information for the current of the 12v battery goes into CAN MO 6
        MOV R2, #05555h         ; This bit pattern deactives MCRs
        MOV R1, #05595h         ; SAVE the Configuration of the MCR
        MOV MCR_M2, R2          ; Turn Off the Message Control Register for message
```

```
object 2
        MOV MCR_M4, R2           ; Turn off MCR4

        ;; The State of Charge of the Battery is also generated Here
                ;; The current measurement must be converted back into the actual curren
t value
                MOV R3, DPP0:soc_36v_low_word   ; The Low byte of the 36v battery soc
                MOV R4, DPP0:soc_36v_high_word  ; The upper byte of the 36v battery soc

                ;; Now we must check to see if the charge is positive or negative
                ;; This can be done for the 36v battery by checking to see if pin P2.0 i
s a one or a zero
                MOV R2, P2
                AND R2, #0001h  ; This isolates the pin P2.0

                CMP R2, #0001h  ; This performs the comparison and sets the Z condition
flag
                JMP cc_NZ, perform_addition_36v ;The battery is charging when the pin is
 logic level low
perform_subtraction_36v: ;The battery is discharging
                SUB R3, R0
                SUBC R4, ZEROS
                JMP continue_data_collection_36v
perform_addition_36v:                           ; the battery is charging
                ADD R3, R0
                ADDC R4, ZEROS

                ;; When this point is reached the SOC should be in registers R3 and R4.
 The total charge for this period
                ;; should be in R0, the current direction should be in R2, and the curre
nt magnitude should be in R7
continue_data_collection_36v:
        MOV DPP0:current_36v, R0
        MOV DPP0:current_direction_36v, R2
        MOV DPP0:soc_36v_high_word, R4
        MOV DPP0:soc_36v_low_word, R3
        MOVB RH2, RL2   ; Move the current direction into the upper byte of R2
        AND R2,  #00F00h        ; Get rid of all but the 3rd nibble
        SHL R2, #04h    ; Move the direction information into the upper nibble
        ADD R2, R0              ; Move the magnitude of the current into R2
        MOV DATA_M2, R2 ; Magnitude and direction information is now put into Message Ob
ject 2
; Move the SOC into the Message Object 4
        MOV DATA_M41, R4        ; Put the high data byte into data registers 2 and 1
        MOV DATA_M42, R3        ; Put the low data byte into data registers 4 and 3

        MOV MCR_M4, R1  ; Restore the SOC Message Object
        MOV MCR_M2, R1  ; Restore the CAN message object to operational status
        CALL energy_management_algorithm
        MOV R9, #04h
        ADD P2, R9

        BSET T3R                ; Start the Conversion Again

        JMP exit_routine

exit_routine:
        POP R9
        POP R8
        POP R7
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        POP DPP2
        POP DPP1
        POP DPP0
        RET
atod_function ENDP
atod_handlers ENDS

END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS


name ema2       ; THIS IS THE ENERGY MANAGEMENT ALGORITHM ASSEMBLY FILE

ASSUME DPP3:SYSTEM
RBANK1 COMREG R0-R15
GLOBAL energy_management_algorithm
GLOBAL dcdcinitialize

EXTERN voltage_36v:WORD
EXTERN current_36v:WORD
EXTERN current_direction_36v:WORD
EXTERN temperature_36v:WORD
EXTERN soc_36v_high_word:WORD
EXTERN soc_36v_low_word:WORD
EXTERN voltage_12v:WORD
EXTERN current_12v:WORD
EXTERN current_direction_12v:WORD
EXTERN temperature_12v:WORD
EXTERN soc_12v_high_word:WORD
EXTERN soc_12v_low_word:WORD
EXTERN soc_region_36v:WORD
EXTERN soc_region_12v:WORD

EXTERN r1_soc_36v_high:WORD
EXTERN r1_soc_36v_low:WORD
EXTERN r2_soc_36v_high:WORD
EXTERN r2_soc_36v_low:WORD
EXTERN r3_soc_36v_high:WORD
EXTERN r3_soc_36v_low:WORD
EXTERN r4_soc_36v_high:WORD
EXTERN r4_soc_36v_low:WORD

EXTERN r1_soc_12v_high:WORD
EXTERN r1_soc_12v_low:WORD
EXTERN r2_soc_12v_high:WORD
EXTERN r2_soc_12v_low:WORD
EXTERN r3_soc_12v_high:WORD
EXTERN r3_soc_12v_low:WORD
EXTERN r4_soc_12v_high:WORD
EXTERN r4_soc_12v_low:WORD

dcdc_data_section SECTION DATA WORD GLOBAL 'RAM'
        dcdc_state      DSW 1
dcdc_data_section ENDS

dcdc_decisions SECTION DATA WORD GLOBAL 'ROM'
        ;; There are 5 decisions to be made
        ;; 0 = NONE
        ;; 1 = Full
        ;; 2 = ZERO
        ;; 3 = UP
        ;; 4 = DOWN
        ;; The hex symbol next to some of the values is unnecessary but was put
        ;; in for test purposes
        decision_11_mm  DW 0
        decision_11_mp  DW 1
        decision_11_pm  DW 4
        decision_11_pp  DW 2

        decision_12_mm  DW 0
        decision_12_mp  DW 3
        decision_12_pm  DW 0
        decision_12_pp  DW 4

        decision_13_mm  DW 2
        decision_13_mp  DW 0
        decision_13_pm  DW 2  ;modified for test purposes real value is 2
        decision_13_pp  DW 2

        decision_14_mm  DW 2
        decision_14_mp  DW 2
        decision_14_pm  DW 2
        decision_14_pp  DW 2

        decision_15_mm  DW 2
        decision_15_mp  DW 2
        decision_15_pm  DW 2
        decision_15_pp  DW 2

        decision_21_mm  DW 3
        decision_21_mp  DW 1
        decision_21_pm  DW 0
        decision_21_pp  DW 1

        decision_22_mm  DW 0
        decision_22_mp  DW 3
        decision_22_pm  DW 4
        decision_22_pp  DW 2

        decision_23_mm  DW 0
        decision_23_mp  DW 0
        decision_23_pm  DW 4
        decision_23_pp  DW 4

        decision_24_mm  DW 2
        decision_24_mp  DW 2
        decision_24_pm  DW 2
        decision_24_pp  DW 2

        decision_25_mm  DW 2
        decision_25_mp  DW 2
        decision_25_pm  DW 2
        decision_25_pp  DW 2

        decision_31_mm  DW 1
        decision_31_mp  DW 1
        decision_31_pm  DW 1
        decision_31_pp  DW 1

        decision_32_mm  DW 1
        decision_32_mp  DW 1
        decision_32_pm  DW 1
        decision_32_pp  DW 1

        decision_33_mm  DW 0
        decision_33_mp  DW 0
        decision_33_pm  DW 0
        decision_33_pp  DW 0

        decision_34_mm  DW 4
        decision_34_mp  DW 4
        decision_34_pm  DW 4
        decision_34_pp  DW 4

        decision_35_mm  DW 2
        decision_35_mp  DW 2
        decision_35_pm  DW 2
        decision_35_pp  DW 2

        decision_41_mm  DW 1
        decision_41_mp  DW 1
        decision_41_pm  DW 1
        decision_41_pp  DW 1

        decision_42_mm  DW 1
        decision_42_mp  DW 1
        decision_42_pm  DW 1
        decision_42_pp  DW 1

        decision_43_mm  DW 3
        decision_43_mp  DW 3
        decision_43_pm  DW 3
        decision_43_pp  DW 3

        decision_44_mm  DW 4
        decision_44_mp  DW 3
        decision_44_pm  DW 4
        decision_44_pp  DW 0

        decision_45_mm  DW 2
        decision_45_mp  DW 3
        decision_45_pm  DW 2
        decision_45_pp  DW 2

        decision_51_mm  DW 2
        decision_51_mp  DW 2
        decision_51_pm  DW 2
        decision_51_pp  DW 2

        decision_52_mm  DW 2
        decision_52_mp  DW 2
        decision_52_pm  DW 2
        decision_52_pp  DW 2

        decision_53_mm  DW 2
        decision_53_mp  DW 2
        decision_53_pm  DW 2
        decision_53_pp  DW 2

        decision_54_mm  DW 4
        decision_54_mp  DW 3
        decision_54_pm  DW 4
        decision_54_pp  DW 3

        decision_55_mm  DW 0
        decision_55_mp  DW 0
        decision_55_pm  DW 0
        decision_55_pp  DW 0
dcdc_decisions ENDS

dcdcstart SECTION CODE
dcdcinitialize PROC FAR
        ;; This function simply initializes the DC/DC converter to ZERO output
        PUSH DPP0
        MOV DPP0, #PAG dcdc_data_section
        NOP
```

```
        MOV DPP0:dcdc_state, ZEROS
        POP DPP0
        RET
dcdcinitialize ENDP
dcdcstart ENDS
energy_management SECTION CODE
energy_management_algorithm PROC FAR
        PUSH R0
        PUSH DPP0
        CALL determine_soc_36v
        CALL determine_soc_12v
        CALL ema_decision
        MOV DPP0, #PAG dcdc_data_section
        NOP
        MOV R0, DPP0:dcdc_state
        MOV DATA_M9, R0
        MOV R0, #06595h ; transmit the data in DATA_M9 which happens to be the wante
d DC/DC converter state
        MOV MCR_M9, R0
        POP DPP0
        POP R0
        RET
energy_management_algorithm ENDP
energy_management ENDS

energy_management_options SECTION CODE
ema_decision PROC FAR
;; This function takes and makes a decision as to what to do about the state of the
DC/DC converter
;; Based on the Region of state of charge of both batteries and their currents
;; It does this by using a giant WORD lookup table.  This WORD is put into the varia
ble
;; dcdcstate1, and from there it is decided what to do with it.
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        PUSH R8
        PUSH R9
        PUSH R10
        PUSH R11
        PUSH MDH
        PUSH MDL
        PUSH DPP0
        PUSH DPP1
        PUSH DPP2
        MOV DPP0, #PAG current_direction_36v
        AND R6, ZEROS   ; This is to be used in looking up the array index.
        AND R7, ZEROS   ; This is to be used as a pointer to our array.
        ;; These are the variables needed to make a decision about the
        ;; State of the DC/DC converter
        MOV R0, DPP0:current_direction_36v
        MOV R1, DPP0:current_direction_12v
        MOV R2, DPP0:soc_region_36v
        MOV R3, DPP0:soc_region_12v
        ;;  The function for computing the memory location to look in is
        ;; The soc_region_12v - 1 = the number of 20s in the offset
        ;; The soc_region_36v -1 = the number of 4s in the offset
        ;; and the current signs gives one of 4 different offsets
        ;; (12,36) => (-,-) = 1 ; (-, +) = 2 ; (+,-) = 3; (+,+) = 4
        ;; Adding them all togther gives you up to 100 different choices
        ;; Subtracting by one gives the appropriate array index
        ;; First determine the number of 20s
        SUB R3, #01h  ; R3 now contains the number of 20s that are in offset index
        ;; Now determine the number of 4s in the index
        SUB R2, #01h ; R2, now contains the number of 4s that are in the index.
        ;; Now Compute the Major index by unsigned multiplication
        MOV R4, #14h
        MULU R3, R4    ; 14h is 20 in hex
        NOP
        MOV R3, MDL                ; Now R3 contains a number between zero and 80
        MOV R4, #4h
        MULU R2, R4      ; 4h is 4 in hex
        NOP
        MOV R2, MDL                ; Now R2 contains a number between zero and 16
        NOP
        ADD R3, R2                 ; Now R3 has the index less the offset of 4 created by t
he current signs.

        ;; Now Determine the offset due to the current direction.
        CMP R1, ZEROS  ; Test the 12v current direction
        JMP cc_Z, plus_12v

minus_12v:
        CMP R0, ZEROS ; Test the 36v current direction
        JMP cc_Z, plus_one_36v
        MOV R5, #01h    ; Negative 36v current direction
        JMP finalize_index

plus_12v:
        CMP R0, ZEROS   ; test the 36v current direction
        JMP cc_Z, plus_two_36v
        MOV R5, #03h
        JMP finalize_index

plus_one_36v:
        MOV R5, #02h
        JMP finalize_index

plus_two_36v:
        MOV R5, #04h

finalize_index:
        MOV R0, #02h
        ADD R3, R5
        MULU R3, R0
        NOP
        MOV R3, MDL

        SUB R3, #02h ;; Now R3 has the final index.  Now the appropriate word can be loo
ked up in our lookup table.

        MOV DPP2, #PAG dcdc_decisions
        NOP
        MOV R8, #DPP2:dcdc_decisions    ; move the address of the first item in the arra
y into register 8
        ADD R8, R3

get_data:
        MOV R9, [R8] ;  This puts the decision of the DC/DC converter into R9

        MOV DPP1, #PAG dcdc_data_section
        NOP
        MOV DPP1:dcdc_state, R9
;; Finally test the 12v battery's voltage
        ;; if it is less than 13v Go to full on
```

```
        MOV DPP0, #PAG voltage_12v
        NOP
        MOV R10, DPP0:voltage_12v
        MOV R11, #03FFh
        CMP R11, R10
        JMP cc_NC, full_on

        CMP R9, ZEROS   ; In this case don't do anything
        JMP cc_Z, exit_dcdc_index
        CMP R9, #01h    ; full on
        JMP cc_Z, full_on
        CMP R9, #02h     ; Full off
        JMP cc_Z, full_off
        CMP R9, #03h     ; Up one
        JMP cc_Z, up_one
        CMP R9, #04h     ; Down one
        JMP cc_Z, down_one
                JMP exit_dcdc_index

full_on:
        MOV DPP1, #PAG dcdc_data_section
        NOP
        MOV DPP1:dcdc_state, ZEROS      ; ZEROS produces full on for the DC/DC conve
rter
        JMP exit_dcdc_index

full_off:
        MOV DPP1, #PAG dcdc_data_section
        NOP
        MOV DPP1:dcdc_state, ONES       ; ONES produces full off for the DC/DC conve
rter
        JMP exit_dcdc_index

up_one:
        MOV DPP1, #PAG dcdc_data_section
        NOP
        MOV R0, DPP1:dcdc_state
        CMPB RL0, #000h ; see if already at max
        JMP cc_Z, exit_dcdc_index
        SUB R0, #01h
        MOV DPP1:dcdc_state, R0         ; New value for the DC/DC converter
        JMP exit_dcdc_index

down_one:
        MOV DPP1, #PAG dcdc_data_section
        NOP
        MOV R0, DPP1:dcdc_state
        CMPB RL0, #0FFh ; see if already at min
        JMP cc_Z, exit_dcdc_index
        ADD R0, #01h
        MOV DPP1:dcdc_state, R0         ; new value for DCDC converter
        JMP exit_dcdc_index


exit_dcdc_index:
        POP DPP2
        POP DPP1
        POP DPP0
        POP MDL
        POP MDH
        POP R11
        POP R10
        POP R9
        POP R8
        POP R7
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
ema_decision ENDP
energy_management_options ENDS



determine_soc_region SECTION CODE

;; This procedure trys to determine which of 5 possible different regions of
;; State of Charge that a battery is operating in.
determine_soc_36v PROC FAR
        PUSH R0
        PUSH R1
        PUSH DPP0
        PUSH DPP1
        MOV DPP0, #PAG soc_36v_high_word
        MOV DPP1, #PAG r1_soc_36v_high
        MOV R0, DPP0:soc_36v_high_word
        MOV R1, DPP1:r1_soc_36v_high
        CMP R1, R0  ; This subtracts soc_36v_high_word from r1_soc_36v_high so then test
 flags

        ; If there is a carry then soc_36v_high_word was larger
        ; than r1_soc_36v_high so a carry was generated
        ; soc_36v_high_word > r1_soc_36v_high => Very Dangerous Over Charge => Region 1
        JMP cc_C, Region1_36v

        ; If no Carry must test to see if soc_36v_high_word = r1_soc_36v_high
        ; If they DON'T equal then soc_36v_high_word < r1_soc_36v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_2_36v

        ; Since soc_36v_high_word = r1_soc_36v_high must now test lower word
        ; Inorder to determine if battery is in region 1 or region 2
        MOV R0, DPP0:soc_36v_low_word
        MOV R1, DPP1:r1_soc_36v_low
        CMP R1, R0      ; This subtracts soc_36v_low_word from r1_soc_36v_low

        ; If soc_36v_low_word > r1_soc_36v_low
        ; then operating in region 1
        JMP cc_C, Region1_36v

        ; If no Carry must test to see if soc_36v_low_word = r1_soc_36v_low
        ; If they DON'T equal then soc_36v_low_word < r1_soc_36v_low
        ; This means region 2
        JMP cc_NZ, Region2_36v

        ; Getting here means that the soc_36v_high_word = r1_soc_36v_high
        ; This point is defined to be in Region 1
        JMP Region1_36v

Test_Region_2_36v:
        MOV R1, DPP1:r2_soc_36v_high
        NOP
        CMP R1, R0  ; This subtracts soc_36v_high_word from r2_soc_36v_high so then test
 flags

        ; If there is a carry then soc_36v_high_word was larger
```

```
        ; than r2_soc_36v_high so a carry was generated
        ; soc_36v_high_word > r2_soc_36v_high => Very Dangerous Over Charge => Regio
n 1
        JMP cc_C, Region2_36v

        ; If no Carry must test to see if soc_36v_high_word = r2_soc_36v_high
        ; If they DON'T equal then soc_36v_high_word < r2_soc_36v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_3_36v

        ; Since soc_36v_high_word = r2_soc_36v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_36v_low_word
        MOV R1, DPP1:r2_soc_36v_low
        CMP R1, R0      ; This subtracts soc_36v_low_word from r2_soc_36v_low

        ; If soc_36v_low_word > r2_soc_36v_low
        ; then operating in region 2
        JMP cc_C, Region2_36v

        ; If no Carry must test to see if soc_36v_low_word = r2_soc_36v_low
        ; If they DON'T equal then soc_36v_low_word < r2_soc_36v_low
        ; This means region 3
        JMP cc_NZ, Region3_36v

        ; Getting here means that the soc_36v_high_word = r2_soc_36v_high
        ; This point is defined to be in Region 2
        JMP Region2_36v

Test_Region_3_36v:
        MOV R1, DPP1:r3_soc_36v_high
        NOP
        CMP R1, R0  ; This subtracts soc_36v_high_word from r3_soc_36v_high so then
test flags

        ; If there is a carry then soc_36v_high_word was larger
        ; than r3_soc_36v_high so a carry was generated
        ; soc_36v_high_word > r3_soc_36v_high => Ideal Operation => Region 3
        JMP cc_C, Region3_36v

        ; If no Carry must test to see if soc_36v_high_word = r3_soc_36v_high
        ; If they DON'T equal then soc_36v_high_word < r3_soc_36v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_4_36v

        ; Since soc_36v_high_word = r3_soc_36v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_36v_low_word
        MOV R1, DPP1:r3_soc_36v_low
        CMP R1, R0      ; This subtracts soc_36v_low_word from r3_soc_36v_low

        ; If soc_36v_low_word > r3_soc_36v_low
        ; then operating in region 2
        JMP cc_C, Region2_36v

        ; If no Carry must test to see if soc_36v_low_word = r3_soc_36v_low
        ; If they DON'T equal then soc_36v_low_word < r3_soc_36v_low
        ; This means region 4
        JMP cc_NZ, Region4_36v

        ; Getting here means that the soc_36v_high_word = r3_soc_36v_high
        ; This point is defined to be in Region 3
        JMP Region3_36v

Test_Region_4_36v:
        MOV R1, DPP1:r4_soc_36v_high
        NOP
        CMP R1, R0  ; This subtracts soc_36v_high_word from r4_soc_36v_high so then test
 flags

        ; If there is a carry then soc_36v_high_word was larger
        ; than r4_soc_36v_high so a carry was generated
        ; soc_36v_high_word > r4_soc_36v_high => Moderate Undercharge => Region 4
        JMP cc_C, Region4_36v

        ; If no Carry must test to see if soc_36v_high_word = r4_soc_36v_high
        ; If they DON'T equal then soc_36v_high_word < r4_soc_36v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_5_36v

        ; Since soc_36v_high_word = r4_soc_36v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_36v_low_word
        MOV R1, DPP1:r4_soc_36v_low
        CMP R1, R0      ; This subtracts soc_36v_low_word from r4_soc_36v_low

        ; If soc_36v_low_word > r4_soc_36v_low
        ; then operating in region 2
        JMP cc_C, Region4_36v

        ; If no Carry must test to see if soc_36v_low_word = r4_soc_36v_low
        ; If they DON'T equal then soc_36v_low_word < r4_soc_36v_low
        ; This means region 2
        JMP cc_NZ, Region5_36v

        ; Getting here means that the soc_36v_high_word = r4_soc_36v_high
        ; This point is defined to be in Region 2
        JMP Region4_36v

Test_Region_5_36v:
        JMP Region5_36v



Region1_36v:
        MOV R0, #01h  ; Move the region number into R0
        MOV DPP0:soc_region_36v, R0     ; Put that number into memory
        JMP exit_soc_36v


Region2_36v:
        MOV R0, #02h  ; Move the region number into R0
        MOV DPP0:soc_region_36v, R0     ; Put that number into memory
        JMP exit_soc_36v


Region3_36v:
        MOV R0, #03h  ; Move the region number into R0
        MOV DPP0:soc_region_36v, R0     ; Put that number into memory
        JMP exit_soc_36v


Region4_36v:
        MOV R0, #04h  ; Move the region number into R0
        MOV DPP0:soc_region_36v, R0     ; Put that number into memory
        JMP exit_soc_36v


Region5_36v:
        MOV R0, #05h  ; Move the region number into R0
```

```
        MOV DPP0:soc_region_36v, R0     ; Put that number into memory
        JMP exit_soc_36v

exit_soc_36v:
        POP DPP1
        POP DPP0
        POP R1
        POP R0
        RET
determine_soc_36v ENDP

determine_soc_12v PROC FAR
        PUSH R0
        PUSH R1
        PUSH DPP0
        PUSH DPP1
        MOV DPP0, #PAG soc_12v_high_word
        MOV DPP1, #PAG r1_soc_12v_high
        MOV R0, DPP0:soc_12v_high_word
        MOV R1, DPP1:r1_soc_12v_high
        CMP R1, R0  ; This subtracts soc_12v_high_word from r1_soc_12v_high so then
test flags

        ; If there is a carry then soc_12v_high_word was larger
        ; than r1_soc_12v_high so a carry was generated
        ; soc_12v_high_word > r1_soc_12v_high => Very Dangerous Over Charge => Regio
n 1
        JMP cc_C, Region1_12v

        ; If no Carry must test to see if soc_12v_high_word = r1_soc_12v_high
        ; If they DON'T equal then soc_12v_high_word < r1_soc_12v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_2_12v

        ; Since soc_12v_high_word = r1_soc_12v_high must now test lower word
        ; Inorder to determine if battery is in region 1 or region 2
        MOV R0, DPP0:soc_12v_low_word
        MOV R1, DPP1:r1_soc_12v_low
        CMP R1, R0      ; This subtracts soc_12v_low_word from r1_soc_12v_low

        ; If soc_12v_low_word > r1_soc_12v_low
        ; then operating in region 1
        JMP cc_C, Region1_12v

        ; If no Carry must test to see if soc_12v_low_word = r1_soc_12v_low
        ; If they DON'T equal then soc_12v_low_word < r1_soc_12v_low
        ; This means region 2
        JMP cc_NZ, Region2_12v

        ; Getting here means that the soc_12v_high_word = r1_soc_12v_high
        ; This point is defined to be in Region 1
        JMP Region1_12v

Test_Region_2_12v:
        MOV R1, DPP1:r2_soc_12v_high
        NOP
        CMP R1, R0  ; This subtracts soc_12v_high_word from r2_soc_12v_high so then
test flags

        ; If there is a carry then soc_12v_high_word was larger
        ; than r2_soc_12v_high so a carry was generated
        ; soc_12v_high_word > r2_soc_12v_high => Very Dangerous Over Charge => Regio
n 1
        JMP cc_C, Region2_12v
        ; If no Carry must test to see if soc_12v_high_word = r2_soc_12v_high
        ; If they DON'T equal then soc_12v_high_word < r2_soc_12v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_3_12v

        ; Since soc_12v_high_word = r2_soc_12v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_12v_low_word
        MOV R1, DPP1:r2_soc_12v_low
        CMP R1, R0      ; This subtracts soc_12v_low_word from r2_soc_12v_low

        ; If soc_12v_low_word > r2_soc_12v_low
        ; then operating in region 2
        JMP cc_C, Region2_12v

        ; If no Carry must test to see if soc_12v_low_word = r2_soc_12v_low
        ; If they DON'T equal then soc_12v_low_word < r2_soc_12v_low
        ; This means region 3
        JMP cc_NZ, Region3_12v

        ; Getting here means that the soc_12v_high_word = r2_soc_12v_high
        ; This point is defined to be in Region 2
        JMP Region2_12v

Test_Region_3_12v:
        MOV R1, DPP1:r3_soc_12v_high
        NOP
        CMP R1, R0  ; This subtracts soc_12v_high_word from r3_soc_12v_high so then test
 flags

        ; If there is a carry then soc_12v_high_word was larger
        ; than r3_soc_12v_high so a carry was generated
        ; soc_12v_high_word > r3_soc_12v_high => Ideal Operation => Region 3
        JMP cc_C, Region3_12v

        ; If no Carry must test to see if soc_12v_high_word = r3_soc_12v_high
        ; If they DON'T equal then soc_12v_high_word < r3_soc_12v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_4_12v

        ; Since soc_12v_high_word = r3_soc_12v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_12v_low_word
        MOV R1, DPP1:r3_soc_12v_low
        CMP R1, R0      ; This subtracts soc_12v_low_word from r3_soc_12v_low

        ; If soc_12v_low_word > r3_soc_12v_low
        ; then operating in region 2
        JMP cc_C, Region2_12v

        ; If no Carry must test to see if soc_12v_low_word = r3_soc_12v_low
        ; If they DON'T equal then soc_12v_low_word < r3_soc_12v_low
        ; This means region 4
        JMP cc_NZ, Region4_12v

        ; Getting here means that the soc_12v_high_word = r3_soc_12v_high
        ; This point is defined to be in Region 3
        JMP Region3_12v

Test_Region_4_12v:
        MOV R1, DPP1:r4_soc_12v_high
        NOP
        CMP R1, R0  ; This subtracts soc_12v_high_word from r4_soc_12v_high so then test
 flags
```

```
        ; If there is a carry then soc_12v_high_word was larger
        ; than r4_soc_12v_high so a carry was generated
        ; soc_12v_high_word > r4_soc_12v_high => Moderate Undercharge => Region 4
        JMP cc_C, Region4_12v

        ; If no Carry must test to see if soc_12v_high_word = r4_soc_12v_high
        ; If they DON'T equal then soc_12v_high_word < r4_soc_12v_high
        ; This means Test for Different Region
        JMP cc_NZ, Test_Region_5_12v

        ; Since soc_12v_high_word = r4_soc_12v_high must now test lower word
        ; Inorder to determine if battery is in region 2 or region 3
        MOV R0, DPP0:soc_12v_low_word
        MOV R1, DPP1:r4_soc_12v_low
        CMP R1, R0      ; This subtracts soc_12v_low_word from r4_soc_12v_low

        ; If soc_12v_low_word > r4_soc_12v_low
        ; then operating in region 2
        JMP cc_C, Region4_12v

        ; If no Carry must test to see if soc_12v_low_word = r4_soc_12v_low
        ; If they DON'T equal then soc_12v_low_word < r4_soc_12v_low
        ; This means region 2
        JMP cc_NZ, Region5_12v

        ; Getting here means that the soc_12v_high_word = r4_soc_12v_high
        ; This point is defined to be in Region 2
        JMP Region4_12v

Test_Region_5_12v:
        JMP Region5_12v

Region1_12v:
        MOV R0, #01h  ; Move the region number into R0
        MOV DPP0:soc_region_12v, R0     ; Put that number into memory
        JMP exit_soc_12v

Region2_12v:
        MOV R0, #02h  ; Move the region number into R0
        MOV DPP0:soc_region_12v, R0     ; Put that number into memory
        JMP exit_soc_12v

Region3_12v:
        MOV R0, #03h  ; Move the region number into R0
        MOV DPP0:soc_region_12v, R0     ; Put that number into memory
        JMP exit_soc_12v

Region4_12v:
        MOV R0, #04h  ; Move the region number into R0
        MOV DPP0:soc_region_12v, R0     ; Put that number into memory
        JMP exit_soc_12v

Region5_12v:
        MOV R0, #05h  ; Move the region number into R0
        MOV DPP0:soc_region_12v, R0     ; Put that number into memory
        JMP exit_soc_12v

exit_soc_12v:
        POP DPP1
        POP DPP0
        POP R1
        POP R0
        RET
determine_soc_12v ENDP
determine_soc_region ENDS
```

```
END
```

```
LOCATE
main.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES('RAM' (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO main.out
```

```
;*********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA    0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA    0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA    0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA    0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M41                DEFA    0EF48h
DATA_M42                DEFA    0EF4Ah
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M81                DEFA    0EF88h
DATA_M82                DEFA    0EF8Ah
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h
```

```
DP8             DEFR    0FFD6h
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
```

```
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
```

```
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     ‘CC0IO’
EX1IN           LIT     ‘CC1IO’
EX2IN           LIT     ‘CC2IO’
EX3IN           LIT     ‘CC3IO’

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
```

```
AN11            DEFB    P5.11
AN12            DEFB    P5.12
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     ‘AN10’
T5EUD           LIT     ‘AN11’
T6IN            LIT     ‘AN12’
T5IN            LIT     ‘AN13’
T4EUD           LIT     ‘AN14’
T2EUD           LIT     ‘AN15’

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15
```

```
ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
ACC22           DEFB    CCM5.11
ACC23           DEFB    CCM5.15

ACC24           DEFB    CCM6.3
ACC25           DEFB    CCM6.7
ACC26           DEFB    CCM6.11
ACC27           DEFB    CCM6.15

ACC28           DEFB    CCM7.3
ACC29           DEFB    CCM7.7
ACC30           DEFB    CCM7.11
ACC31           DEFB    CCM7.15

T2R             DEFB    T2CON.6
T2UD            DEFB    T2CON.7
T2UDE           DEFB    T2CON.8

T3R             DEFB    T3CON.6
T3UD            DEFB    T3CON.7
T3UDE           DEFB    T3CON.8
T3OE            DEFB    T3CON.9
T3OTL           DEFB    T3CON.10

T4R             DEFB    T4CON.6
T4UD            DEFB    T4CON.7
T4UDE           DEFB    T4CON.8

T5R             DEFB    T5CON.6
T5UD            DEFB    T5CON.7
T5UDE           DEFB    T5CON.8
T5CLR           DEFB    T5CON.14
T5SC            DEFB    T5CON.15

T6R             DEFB    T6CON.6
T6UD            DEFB    T6CON.7
T6UDE           DEFB    T6CON.8
T6OE            DEFB    T6CON.9
T6OTL           DEFB    T6CON.10
T6SR            DEFB    T6CON.15

T2IE            DEFB    T2IC.6
T2IR            DEFB    T2IC.7
T3IE            DEFB    T3IC.6
T3IR            DEFB    T3IC.7
T4IE            DEFB    T4IC.6
T4IR            DEFB    T4IC.7
T5IE            DEFB    T5IC.6
T5IR            DEFB    T5IC.7
T6IE            DEFB    T6IC.6
T6IR            DEFB    T6IC.7

CRIE            DEFB    CRIC.6
CRIR            DEFB    CRIC.7

S0TIE           DEFB    S0TIC.6
S0TIR           DEFB    S0TIC.7
S0RIE           DEFB    S0RIC.6
S0RIR           DEFB    S0RIC.7
S0EIE           DEFB    S0EIC.6
S0EIR           DEFB    S0EIC.7
S0TBIE          DEFB    S0TBIC.6
S0TBIR          DEFB    S0TBIC.7
```

```
SSCTIE          DEFB    SSCTIC.6
SSCTIR          DEFB    SSCTIC.7
SSCRIE          DEFB    SSCRIC.6
SSCRIR          DEFB    SSCRIC.7
SSCEIE          DEFB    SSCEIC.6
SSCEIR          DEFB    SSCEIC.7
SSCTE           LIT     'SSCTEN'
SSCRE           LIT     'SSCREN'
SSCPE           LIT     'SSCPEN'
SSCBE           LIT     'SSCBEN'


CC0IE           DEFB    CC0IC.6
CC0IR           DEFB    CC0IC.7
CC1IE           DEFB    CC1IC.6
CC1IR           DEFB    CC1IC.7
CC2IE           DEFB    CC2IC.6
CC2IR           DEFB    CC2IC.7
CC3IE           DEFB    CC3IC.6
CC3IR           DEFB    CC3IC.7
CC4IE           DEFB    CC4IC.6
CC4IR           DEFB    CC4IC.7
CC5IE           DEFB    CC5IC.6
CC5IR           DEFB    CC5IC.7
CC6IE           DEFB    CC6IC.6
CC6IR           DEFB    CC6IC.7
CC7IE           DEFB    CC7IC.6
CC7IR           DEFB    CC7IC.7
CC8IE           DEFB    CC8IC.6
CC8IR           DEFB    CC8IC.7
CC9IE           DEFB    CC9IC.6
CC9IR           DEFB    CC9IC.7
CC10IE          DEFB    CC10IC.6
CC10IR          DEFB    CC10IC.7
CC11IE          DEFB    CC11IC.6
CC11IR          DEFB    CC11IC.7
CC12IE          DEFB    CC12IC.6
CC12IR          DEFB    CC12IC.7
CC13IE          DEFB    CC13IC.6
CC13IR          DEFB    CC13IC.7
CC14IE          DEFB    CC14IC.6
CC14IR          DEFB    CC14IC.7
CC15IE          DEFB    CC15IC.6
CC15IR          DEFB    CC15IC.7
CC16IE          DEFB    CC16IC.6
CC16IR          DEFB    CC16IC.7
CC17IE          DEFB    CC17IC.6
CC17IR          DEFB    CC17IC.7
CC18IE          DEFB    CC18IC.6
CC18IR          DEFB    CC18IC.7
CC19IE          DEFB    CC19IC.6
CC19IR          DEFB    CC19IC.7
CC20IE          DEFB    CC20IC.6
CC20IR          DEFB    CC20IC.7
CC21IE          DEFB    CC21IC.6
CC21IR          DEFB    CC21IC.7
CC22IE          DEFB    CC22IC.6
CC22IR          DEFB    CC22IC.7
CC23IE          DEFB    CC23IC.6
CC23IR          DEFB    CC23IC.7
CC24IE          DEFB    CC24IC.6
CC24IR          DEFB    CC24IC.7
CC25IE          DEFB    CC25IC.6
CC25IR          DEFB    CC25IC.7
CC26IE          DEFB    CC26IC.6
```

```
CC26IR          DEFB    CC26IC.7
CC27IE          DEFB    CC27IC.6
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15

PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

```
a166 main.asm
a166 canmod.asm
a166 canmo.asm
a166 canint.asm
l166 LINK main.obj canmod.obj canmo.obj canint.obj TO main.lno
l166 @linker.lnv
ihex166 -i16 main.out -o main.hex
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTSSK                         ; CAN USE ALL internal RAM for Stack
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME main
RBANK1  COMREG R0-R15           ; define a common register area of 16 register

SSKDEF 4                        ; default stack size of 256 Words

ASSUME DPP3:SYSTEM

EXTERN canin:FAR                ; Can function


mainseg SECTION CODE
        main PROC FAR

        start:  DISWDT          ; disable the watchdog timer
                BSET IEN        ; Globally Enable Interrupts both global

        ;; Initialize the External Memory BUS
                MOV SYSCON, #0E084h
                MOV ADDRSEL1, #0404h
                MOV BUSCON0, #004AFh
                MOV BUSCON1, #004AFh
                EINIT           ; end initialization
        ;; End of external memory bus initialization

        ;; Initialize the Data Page pointers for this section
                MOV DPP3, #03h           ; make DPP3 point to system
        ;; End of Data Page Pointer Initialization




        ;; Make sure Port 2 is in Open Drain mode
                MOV ODP2, ONES
        ;; Make the direction of Port 2 to output
                MOV DP2, ONES
        ;; Make sure all of the ports are off
                MOV P2, ONES
                BCLR P2.8

        ;; Initialize The Stack
        ;; The Stack pointers are all word pointers so even though the
        ;; highest byte in the stack is located at #0FBFFh the highest
        ;; byte that the stack pointers can point to is #0FBFEh
                MOV STKUN, #0FBFEh; Set Stack Underflow Pointer
                MOV STKOV, #0F800h; Set STack Overflow Pointer
                MOV SP, #0FBFEh ; Set the Stack Pointer
        ;; End of Stack Initialization

        ;; Initialize CAN Bus
              CALL canin        ; Call the CAN initialization function
        ;; End of CAN Bus Initialization

        meto:
                NOP             ; just loop here waiting
                NOP
                JMP meto
```

```
                RET     ; return
main ENDP
mainseg ENDS

startupsec  SECTION CODE        ; codesegment that contains reset int pointer
sysreset PROC TASK INTNO=0H     ; reset interrupt number is zero at 0h
        ORG 000H                ; forces next instruction to be located at 0h
        JMP start               ; installs a pointer to the startup routine
        RETI                    ; return from interrupt
sysreset ENDP
startupsec ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmod

RBANK1  COMREG R0-R15          ; define a common register area of 16 registers
GLOBAL  canin                  ; The function must be declared Global at the
                               ; beginning of the module

EXTERN  canmocfg:FAR          ; configures specific Message objects

ASSUME DPP3:SYSTEM

canfunc  SECTION CODE          ; codesegment that contains reset int pointer

canin   PROC FAR
        PUSH R0
        PUSH R1

        ;; set all of the CAN control registers
        AND C1CSR,ZEROS   ; set control register to zero
        MOV R1, #0043h          ; Set IE and INIT bits
        OR C1CSR,R1     ; set control register to R1's value

        AND C1BTR, ZEROS  ; set Bit timing register to zero
        MOV R1, #03447h         ; set for 125k operation
        OR C1BTR, R1    ; set Bit timing register parameters

        AND C1GMS, ZEROS  ; set Global Mask short register to zero
        MOV R1, #0FFFFh         ; EOFF is what DAVE initialize
        OR C1GMS, R1    ; set GMS

        AND C1UGML, ZEROS ; set Upper global mask long to zero
        MOV R1, #0FFFFh
        OR C1UGML, R1

        MOV R1, #0F8FFh
        AND C1LGML, ZEROS
        OR C1LGML, R1           ; lower global mask

        AND C1UMLM, ZEROS
        OR C1UMLM, R1           ; upper mask of last register
        AND C1LMLM, ZEROS
        OR C1LMLM, R1           ; lower mask of last register

        CALL setall             ; sets all of the CAN registers to off

        CALL canmocfg           ; Configures specific Message Objects

        ;; Setup CAN interrupt and Initialize CAN module
      EXTR #4
        AND XP0IC, ZEROS  ; configure CAN interrupt control Register
        AND R0,ZEROS
        OR R0,#0073h            ; enable interrupt, level is 10 group is 2
        OR XP0IC,R0     ; Configure CAN interrupt Control Register
        AND R1, ZEROS
        OR R1, #00041h  ; crashes if I clear the CPU access to the BTR
        XOR C1CSR, R1   ; end initialize CAN interrupt
        POP R1
        POP R0
```

```
        RET
canin   ENDP

setall PROC FAR                ; This Procedure sets all of the Mess objs invalid
        ;; by using a counter it counts up to 15 and initializes all of the message
        ;; objects along the way.
        PUSH R2
        PUSH R4
        PUSH R5
        AND R5,ZEROS
        OR  R5, #01h            ; Set counter to 1 for first MO
        AND R2,ZEROS
        OR R2,#0EF10h           ; Set pointer to MO1
        AND R4, ZEROS
        OR R4, #5555h           ; Set R4 to make MObs invalid

nextreg:MOV [R2],R4             ; make all message objects invalid
        ADD R2,#10h
        CMPI1 R5,#0Fh
        JMPA CC_NZ,nextreg      ;
        POP R5
        POP R4
        POP R2
        RET
setall ENDP

canfunc ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canmo
RBANK1 COMREG R0-R15               ; declare bank of 16 global registers
GLOBAL canmocfg


can_module      SECTION CODE

ASSUME DPP3:SYSTEM

canmocfg  PROC FAR
        PUSH R1
        PUSH R2
        PUSH R3
        ;; Now set specific CAN control Registers
        ;; initialize message object 1
        ;; initializing this object to be invalid does or removing the code until
        ;; the comment "Setup CAN interrupt and Initialize ...." does
        ;; nothing to prevent the occurrance of the interrupt for the CAN system
        MOV R2, #MCR_M1            ; start of Message Object 1
      AND R1, ZEROS
        OR R1, #5599h              ; Generate a Receive Interrupt if this message object ac
tivates
        MOV [R2],R1     ; set MO1's Control register

        ADD R2,#2h                 ; point to Upper Arbitration register
      AND R3, ZEROS                ; set R3 to
      OR R3, #000Eh              ; message id for message object 1
        MOV [R2],R3                ; message id = #000Eh
        ADD R2, #2h                ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS            ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h              ; put 00h into first data byte and set to receive
      MOV MCD_M1,R1                ; Databyte(0) = 0 and Set to receive and 3 bytes of data
        MOV DATA_M1, ZEROS         ; fill the Data of the MO with Zeros


        ;; Initialize Message Object 2
        ;; This message object receives information about turning the DC/DC converter on
 and off
        ;; For the purpose of the thesis the DC/DC was just left on all the time.
        MOV R2, #MCR_M2            ; start of Message Object 2
      AND R1, ZEROS
        OR R1, #5599h              ; RECEIVE INTERRUPT NOT enabled
        MOV [R2],R1     ; set MO3's Control register
        ADD R2,#2h                 ; point to Upper Arbitration register
        AND R3, ZEROS                ; set R6 to zero
        OR R3, #0021h            ; The number is the Message ID for Message Object 3
        MOV [R2],R3                ; message id = 00021h
        ADD R2, #2h                ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS            ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0030h              ; put 00030h into first data byte and set to receive
        MOV MCD_M2,R1                ; Databyte(0) = 0 and Set to transmit and 3 bytes of d
ata
        MOV DATA_M2, ZEROS         ; Fill the Data of the MO with Zeros

        ;; Initialize Message Object 3
        ;; This message object transmits the present state of the DC/DC converter

        MOV R2, #MCR_M3            ; start of Message Object 3
        AND R1, ZEROS
        OR R1, #5595h              ; RECEIVE INTERRUPT NOT enabled
        MOV [R2],R1     ; set MO2's Control register
        ADD R2,#2h                 ; point to Upper Arbitration register
        AND R3, ZEROS                ; set R6 to zero
        OR R3, #000Fh            ; The number is the Message ID for Message Object 2
        MOV [R2],R3                ; message id = 000F
        ADD R2, #2h                ; Point to the Lower Arbitration Register
        MOV [R2], ZEROS            ; standard Message object so lowerarb = 0h
        AND R1, ZEROS
        OR R1, #0038h              ; put 00038h into first data byte and set to transmi
t
        MOV MCD_M3,R1                ; Databyte(0) = 0 and Set to transmit and 3 bytes
of data
        MOV DATA_M3, ZEROS         ; Fill the Data of the MO with Zeros



        POP R3
        POP R2
        POP R1
        RET
canmocfg ENDP
can_module ENDS
END
```

```
$SEGMENTED
$EXTEND
$EXTSFR
$EXTMEM
$NOMOD166
$STDNAMES(reg167b.def)
$SYMBOLS

NAME canint
RBANK1 COMREG R0-R15              ; declare bank of 16 global registers


ASSUME DPP3:SYSTEM

can_interrupts  SECTION CODE

can_receive_interrupt PROC TASK INTNO=040h
        ORG 0100h
        CALL can_receive_interrupt_handler
        RETI
can_receive_interrupt ENDP

can_receive_interrupt_handler PROC FAR
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        MOVB RL0, INTID           ; Read the CAN interrupt ID buffer
        CMPB RL0, #03h            ; See if the interrupt came from M01
        JMP cc_Z, message_one_interrupt; if interrupt from M01 handle

        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M2, R1
        MOV R0, DATA_M2
        MOV R3, R0                        ; Put the Data in R3 for future use
        MOV MCR_M2, R2

        CMP R0, #01h
        JMP cc_NZ, turn_off_converter
        ;; This is where the converter is turned on
        MOV R4, P2
        BSET R4.8
        MOV P2, R4
        JMP exit_function

turn_off_converter:
        CMP R0, #0800h
        JMP cc_NZ, exit_function
        MOV R4, P2
        BCLR R4.8
        MOV P2, R4
        JMP exit_function

message_one_interrupt:
        ;; Message Object one deals with the state of the DC/DC converter
        MOV R1, #05555h
        MOV R2, #05599h
        MOV MCR_M1, R1
        MOV R0, DATA_M1
        MOV MCR_M1, R2

        ;; Now setup M3 so it can respond to queries about
        ;; the state of the converter
        MOV R2, MCR_M3
        MOV MCR_M3, R1
        MOV DATA_M3, R0
        MOV MCR_M3, R2
        MOV R3, DATA_M3
        MOV R4, P2
        MOVB RL4, RL3
        MOV P2, R4      ; This is where the DC/DC converter is actually set.

exit_function:
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET
can_receive_interrupt_handler ENDP

can_interrupts ENDS
END
```

```
LOCATE
main.lno
{GENERAL}
IRAMSIZE (2048)
RESERVE MEMORY(0F200h TO 0F5FFh)
MEMORY(ROM (0000h to 0EFFFh),
RAM (040000h to 4EFFFh), IRAM(0F000h))
CLASSES('RAM' (040000h to 04FFFFh) )
SYMBOLS LISTSYMBOLS
TO main.out
```

```
;**********************************************************************
;** @(#)reg167b.def     1.10 12/18/97
;**
;** Register definitions for the SAB C167
;** This file contains all SFR names and BIT names
;** This file can be supplied to rm166 and a166 (STDNAMES control)
;*********************************************************************
TRUE            DEFB    0FF20h.0, RW
NODE142         DEFB    0FF20h.1, RW

C1CSR           DEFA    0EF00h
INTID           DEFA    0EF02h
C1BTR           DEFA    0EF04h
C1GMS           DEFA    0EF06h
C1UGML          DEFA    0EF08h
C1LGML          DEFA    0EF0Ah
C1UMLM          DEFA    0EF0Ch
C1LMLM          DEFA    0EF0Eh
MCR_M1          DEFA    0EF10h
MCR_M2          DEFA  0EF20h
MCR_M3          DEFA    0EF30h
MCR_M4          DEFA    0EF40h
MCR_M5          DEFA    0EF50h
MCR_M6          DEFA  0EF60h
MCR_M7          DEFA    0EF70h
MCR_M8          DEFA    0EF80h
MCR_M9          DEFA  0EF90h
MCR_MA          DEFA    0EFA0h
MCR_MB          DEFA    0EFB0h
MCR_MC          DEFA    0EFC0h
MCR_MD          DEFA    0EFD0h
MCR_ME          DEFA    0EFE0h
MCR_MF          DEFA    0EFF0h
MCD_M1          DEFA  0EF16h
MCD_M2          DEFA    0EF26h
MCD_M3          DEFA    0EF36h
MCD_M4          DEFA    0EF46h
MCD_M5          DEFA    0EF56h
MCD_M6          DEFA    0EF66h
MCD_M7          DEFA    0EF76h
MCD_M8          DEFA    0EF86h
MCD_M9          DEFA    0EF96h
MCD_MA          DEFA    0EFA6h
MCD_MB          DEFA    0EFB6h
MCD_MC          DEFA    0EFC6h
MCD_MD          DEFA    0EFD6h
MCD_ME          DEFA    0EFE6h
DATA_M1         DEFA    0EF18h
DATA_M2         DEFA    0EF28h
DATA_M3         DEFA    0EF38h
DATA_M4         DEFA    0EF48h
DATA_M5         DEFA    0EF58h
DATA_M6         DEFA    0EF68h
DATA_M7         DEFA    0EF78h
DATA_M8         DEFA    0EF88h
DATA_M9         DEFA    0EF98h
DATA_MA         DEFA    0EFA8h
DATA_MB         DEFA    0EFB8h
DATA_MC         DEFA    0EFC8h
DATA_MD         DEFA    0EFD8h
DATA_ME         DEFA    0EFE8h



DP8             DEFR    0FFD6h
```

```
P8              DEFR    0FFD4h
DP7             DEFR    0FFD2h
P7              DEFR    0FFD0h
DP6             DEFR    0FFCEh
P6              DEFR    0FFCCh
DP4             DEFR    0FFCAh
P4              DEFR    0FFC8h
DP3             DEFR    0FFC6h
P3              DEFR    0FFC4h
DP2             DEFR    0FFC2h
P2              DEFR    0FFC0h
SSCCON          DEFR    0FFB2h
S0CON           DEFR    0FFB0h
WDTCON          DEFR    0FFAEh
TFR             DEFR    0FFACh
P5              DEFR    0FFA2h
ADCON           DEFR    0FFA0h
T1IC            DEFR    0FF9Eh
T0IC            DEFR    0FF9Ch
ADEIC           DEFR    0FF9Ah
ADCIC           DEFR    0FF98h
CC15IC          DEFR    0FF96h
CC14IC          DEFR    0FF94h
CC13IC          DEFR    0FF92h
CC12IC          DEFR    0FF90h
CC11IC          DEFR    0FF8Eh
CC10IC          DEFR    0FF8Ch
CC9IC           DEFR    0FF8Ah
CC8IC           DEFR    0FF88h
CC7IC           DEFR    0FF86h
CC6IC           DEFR    0FF84h
CC5IC           DEFR    0FF82h
CC4IC           DEFR    0FF80h
CC3IC           DEFR    0FF7Eh
CC2IC           DEFR    0FF7Ch
CC1IC           DEFR    0FF7Ah
CC0IC           DEFR    0FF78h
SSCEIC          DEFR    0FF76h
SSCRIC          DEFR    0FF74h
SSCTIC          DEFR    0FF72h
S0EIC           DEFR    0FF70h
S0RIC           DEFR    0FF6Eh
S0TIC           DEFR    0FF6Ch
CRIC            DEFR    0FF6Ah
T6IC            DEFR    0FF68h
T5IC            DEFR    0FF66h
T4IC            DEFR    0FF64h
T3IC            DEFR    0FF62h
T2IC            DEFR    0FF60h
CCM3            DEFR    0FF58h
CCM2            DEFR    0FF56h
CCM1            DEFR    0FF54h
CCM0            DEFR    0FF52h
T01CON          DEFR    0FF50h
T6CON           DEFR    0FF48h
T5CON           DEFR    0FF46h
T4CON           DEFR    0FF44h
T3CON           DEFR    0FF42h
T2CON           DEFR    0FF40h
PWMCON1         DEFR    0FF32h
PWMCON0         DEFR    0FF30h
CCM7            DEFR    0FF28h
CCM6            DEFR    0FF26h
CCM5            DEFR    0FF24h
CCM4            DEFR    0FF22h
```

```
T78CON          DEFR    0FF20h
P1H             DEFR    0FF06h
P1L             DEFR    0FF04h
P0H             DEFR    0FF02h
P0L             DEFR    0FF00h
PECC7           DEFR    0FECEh
PECC6           DEFR    0FECCh
PECC5           DEFR    0FECAh
PECC4           DEFR    0FEC8h
PECC3           DEFR    0FEC6h
PECC2           DEFR    0FEC4h
PECC1           DEFR    0FEC2h
PECC0           DEFR    0FEC0h
SRCP0           DEFA    0FCE0h
DSTP0           DEFA    0FCE2h
SRCP1           DEFA    0FCE4h
DSTP1           DEFA    0FCE6h
SRCP2           DEFA    0FCE8h
DSTP2           DEFA    0FCEAh
SRCP3           DEFA    0FCECh
DSTP3           DEFA    0FCEEh
SRCP4           DEFA    0FCF0h
DSTP4           DEFA    0FCF2h
SRCP5           DEFA    0FCF4h
DSTP5           DEFA    0FCF6h
SRCP6           DEFA    0FCF8h
DSTP6           DEFA    0FCFAh
SRCP7           DEFA    0FCFCh
DSTP7           DEFA    0FCFEh
S0BG            DEFR    0FEB4h
S0RBUF          DEFR    0FEB2h, r
S0TBUF          DEFR    0FEB0h, w
WDT             DEFR    0FEAEh, r
ADDAT           DEFR    0FEA0h
CC15            DEFR    0FE9Eh
CC14            DEFR    0FE9Ch
CC13            DEFR    0FE9Ah
CC12            DEFR    0FE98h
CC11            DEFR    0FE96h
CC10            DEFR    0FE94h
CC9             DEFR    0FE92h
CC8             DEFR    0FE90h
CC7             DEFR    0FE8Eh
CC6             DEFR    0FE8Ch
CC5             DEFR    0FE8Ah
CC4             DEFR    0FE88h
CC3             DEFR    0FE86h
CC2             DEFR    0FE84h
CC1             DEFR    0FE82h
CC0             DEFR    0FE80h
CC31            DEFR    0FE7Eh
CC30            DEFR    0FE7Ch
CC29            DEFR    0FE7Ah
CC28            DEFR    0FE78h
CC27            DEFR    0FE76h
CC26            DEFR    0FE74h
CC25            DEFR    0FE72h
CC24            DEFR    0FE70h
CC23            DEFR    0FE6Eh
CC22            DEFR    0FE6Ch
CC21            DEFR    0FE6Ah
CC20            DEFR    0FE68h
CC19            DEFR    0FE66h
CC18            DEFR    0FE64h
CC17            DEFR    0FE62h
CC16            DEFR    0FE60h
T1REL           DEFR    0FE56h
T0REL           DEFR    0FE54h
T1              DEFR    0FE52h
T0              DEFR    0FE50h
CAPREL          DEFR    0FE4Ah
T6              DEFR    0FE48h
T5              DEFR    0FE46h
T4              DEFR    0FE44h
T3              DEFR    0FE42h
T2              DEFR    0FE40h
PW3             DEFR    0FE36h
PW2             DEFR    0FE34h
PW1             DEFR    0FE32h
PW0             DEFR    0FE30h

; Extended sfr area

ODP8            DEFR    0F1D6h
ODP7            DEFR    0F1D2h
ODP6            DEFR    0F1CEh
ODP3            DEFR    0F1C6h
PICON           DEFR    0F1C4h
ODP2            DEFR    0F1C2h
EXICON          DEFR    0F1C0h
S0TBIC          DEFR    0F19Ch
XP3IC           DEFR    0F19Eh
XP2IC           DEFR    0F196h
XP1IC           DEFR    0F18Eh
XP0IC           DEFR    0F186h
PWMIC           DEFR    0F17Eh
T8IC            DEFR    0F17Ch
T7IC            DEFR    0F17Ah
CC31IC          DEFR    0F194h
CC30IC          DEFR    0F18Ch
CC29IC          DEFR    0F184h
CC28IC          DEFR    0F178h
CC27IC          DEFR    0F176h
CC26IC          DEFR    0F174h
CC25IC          DEFR    0F172h
CC24IC          DEFR    0F170h
CC23IC          DEFR    0F16Eh
CC22IC          DEFR    0F16Ch
CC21IC          DEFR    0F16Ah
CC20IC          DEFR    0F168h
CC19IC          DEFR    0F166h
CC18IC          DEFR    0F164h
CC17IC          DEFR    0F162h
CC16IC          DEFR    0F160h
RP0H            DEFR    0F108h
DP1H            DEFR    0F106h
DP1L            DEFR    0F104h
DP0H            DEFR    0F102h
DP0L            DEFR    0F100h
SSCBR           DEFR    0F0B4h
SSCRB           DEFR    0F0B2h
SSCTB           DEFR    0F0B0h
ADDAT2          DEFR    0F0A0h
T8REL           DEFR    0F056h
T7REL           DEFR    0F054h
T8              DEFR    0F052h
T7              DEFR    0F050h
PP3             DEFR    0F03Eh
PP2             DEFR    0F03Ch
PP1             DEFR    0F03Ah
```

```
PP0             DEFR    0F038h
PT3             DEFR    0F036h
PT2             DEFR    0F034h
PT1             DEFR    0F032h
PT0             DEFR    0F030h

; Bit names
CC0IO           DEFB    P2.0
CC1IO           DEFB    P2.1
CC2IO           DEFB    P2.2
CC3IO           DEFB    P2.3
CC4IO           DEFB    P2.4
CC5IO           DEFB    P2.5
CC6IO           DEFB    P2.6
CC7IO           DEFB    P2.7
CC8IO           DEFB    P2.8
CC9IO           DEFB    P2.9
CC10IO          DEFB    P2.10
CC11IO          DEFB    P2.11
CC12IO          DEFB    P2.12
CC13IO          DEFB    P2.13
CC14IO          DEFB    P2.14
CC15IO          DEFB    P2.15
EX0IN           LIT     ‘CC0IO’
EX1IN           LIT     ‘CC1IO’
EX2IN           LIT     ‘CC2IO’
EX3IN           LIT     ‘CC3IO’

T0IN            DEFB    P3.0
T6OUT           DEFB    P3.1
CAPIN           DEFB    P3.2
T3OUT           DEFB    P3.3
T3EUD           DEFB    P3.4
T2IN            DEFB    P3.7
T3IN            DEFB    P3.6
T4IN            DEFB    P3.5
SSDI            DEFB    P3.8
SSDO            DEFB    P3.9
TXD0            DEFB    P3.10
RXD0            DEFB    P3.11
SSCLK           DEFB    P3.13
CLKOUT          DEFB    P3.15

A16             DEFB    P4.0
A17             DEFB    P4.1
A18             DEFB    P4.2
A19             DEFB    P4.3
A20             DEFB    P4.4
A21             DEFB    P4.5
A22             DEFB    P4.6
A23             DEFB    P4.7

AN0             DEFB    P5.0
AN1             DEFB    P5.1
AN2             DEFB    P5.2
AN3             DEFB    P5.3
AN4             DEFB    P5.4
AN5             DEFB    P5.5
AN6             DEFB    P5.6
AN7             DEFB    P5.7
AN8             DEFB    P5.8
AN9             DEFB    P5.9
AN10            DEFB    P5.10
AN11            DEFB    P5.11
AN12            DEFB    P5.12
```

```
AN13            DEFB    P5.13
AN14            DEFB    P5.14
AN15            DEFB    P5.15
T6EUD           LIT     ’AN10’
T5EUD           LIT     ’AN11’
T6IN            LIT     ’AN12’
T5IN            LIT     ’AN13’
T4EUD           LIT     ’AN14’
T2EUD           LIT     ’AN15’

POUT0           DEFB    P7.0
POUT1           DEFB    P7.1
POUT2           DEFB    P7.2
POUT3           DEFB    P7.3
CC28IO          DEFB    P7.4
CC29IO          DEFB    P7.5
CC30IO          DEFB    P7.6
CC31IO          DEFB    P7.7

CC16IO          DEFB    P8.0
CC17IO          DEFB    P8.1
CC18IO          DEFB    P8.2
CC19IO          DEFB    P8.3
CC20IO          DEFB    P8.4
CC21IO          DEFB    P8.5
CC22IO          DEFB    P8.6
CC23IO          DEFB    P8.7


T0M             DEFB    T01CON.3
T0R             DEFB    T01CON.6
T1M             DEFB    T01CON.11
T1R             DEFB    T01CON.14
T7M             DEFB    T78CON.3
T7R             DEFB    T78CON.6
T8M             DEFB    T78CON.11
T8R             DEFB    T78CON.14

ACC0            DEFB    CCM0.3
ACC1            DEFB    CCM0.7
ACC2            DEFB    CCM0.11
ACC3            DEFB    CCM0.15

ACC4            DEFB    CCM1.3
ACC5            DEFB    CCM1.7
ACC6            DEFB    CCM1.11
ACC7            DEFB    CCM1.15

ACC8            DEFB    CCM2.3
ACC9            DEFB    CCM2.7
ACC10           DEFB    CCM2.11
ACC11           DEFB    CCM2.15

ACC12           DEFB    CCM3.3
ACC13           DEFB    CCM3.7
ACC14           DEFB    CCM3.11
ACC15           DEFB    CCM3.15

ACC16           DEFB    CCM4.3
ACC17           DEFB    CCM4.7
ACC18           DEFB    CCM4.11
ACC19           DEFB    CCM4.15

ACC20           DEFB    CCM5.3
ACC21           DEFB    CCM5.7
```

```
ACC22            DEFB    CCM5.11
ACC23            DEFB    CCM5.15

ACC24            DEFB    CCM6.3
ACC25            DEFB    CCM6.7
ACC26            DEFB    CCM6.11
ACC27            DEFB    CCM6.15

ACC28            DEFB    CCM7.3
ACC29            DEFB    CCM7.7
ACC30            DEFB    CCM7.11
ACC31            DEFB    CCM7.15

T2R              DEFB    T2CON.6
T2UD             DEFB    T2CON.7
T2UDE            DEFB    T2CON.8

T3R              DEFB    T3CON.6
T3UD             DEFB    T3CON.7
T3UDE            DEFB    T3CON.8
T3OE             DEFB    T3CON.9
T3OTL            DEFB    T3CON.10

T4R              DEFB    T4CON.6
T4UD             DEFB    T4CON.7
T4UDE            DEFB    T4CON.8

T5R              DEFB    T5CON.6
T5UD             DEFB    T5CON.7
T5UDE            DEFB    T5CON.8
T5CLR            DEFB    T5CON.14
T5SC             DEFB    T5CON.15

T6R              DEFB    T6CON.6
T6UD             DEFB    T6CON.7
T6UDE            DEFB    T6CON.8
T6OE             DEFB    T6CON.9
T6OTL            DEFB    T6CON.10
T6SR             DEFB    T6CON.15

T2IE             DEFB    T2IC.6
T2IR             DEFB    T2IC.7
T3IE             DEFB    T3IC.6
T3IR             DEFB    T3IC.7
T4IE             DEFB    T4IC.6
T4IR             DEFB    T4IC.7
T5IE             DEFB    T5IC.6
T5IR             DEFB    T5IC.7
T6IE             DEFB    T6IC.6
T6IR             DEFB    T6IC.7

CRIE             DEFB    CRIC.6
CRIR             DEFB    CRIC.7

S0TIE            DEFB    S0TIC.6
S0TIR            DEFB    S0TIC.7
S0RIE            DEFB    S0RIC.6
S0RIR            DEFB    S0RIC.7
S0EIE            DEFB    S0EIC.6
S0EIR            DEFB    S0EIC.7
S0TBIE           DEFB    S0TBIC.6
S0TBIR           DEFB    S0TBIC.7

SSCTIE           DEFB    SSCTIC.6
SSCTIR           DEFB    SSCTIC.7
```

```
SSCRIE           DEFB    SSCRIC.6
SSCRIR           DEFB    SSCRIC.7
SSCEIE           DEFB    SSCEIC.6
SSCEIR           DEFB    SSCEIC.7
SSCTE            LIT     'SSCTEN'
SSCRE            LIT     'SSCREN'
SSCPE            LIT     'SSCPEN'
SSCBE            LIT     'SSCBEN'


CC0IE            DEFB    CC0IC.6
CC0IR            DEFB    CC0IC.7
CC1IE            DEFB    CC1IC.6
CC1IR            DEFB    CC1IC.7
CC2IE            DEFB    CC2IC.6
CC2IR            DEFB    CC2IC.7
CC3IE            DEFB    CC3IC.6
CC3IR            DEFB    CC3IC.7
CC4IE            DEFB    CC4IC.6
CC4IR            DEFB    CC4IC.7
CC5IE            DEFB    CC5IC.6
CC5IR            DEFB    CC5IC.7
CC6IE            DEFB    CC6IC.6
CC6IR            DEFB    CC6IC.7
CC7IE            DEFB    CC7IC.6
CC7IR            DEFB    CC7IC.7
CC8IE            DEFB    CC8IC.6
CC8IR            DEFB    CC8IC.7
CC9IE            DEFB    CC9IC.6
CC9IR            DEFB    CC9IC.7
CC10IE           DEFB    CC10IC.6
CC10IR           DEFB    CC10IC.7
CC11IE           DEFB    CC11IC.6
CC11IR           DEFB    CC11IC.7
CC12IE           DEFB    CC12IC.6
CC12IR           DEFB    CC12IC.7
CC13IE           DEFB    CC13IC.6
CC13IR           DEFB    CC13IC.7
CC14IE           DEFB    CC14IC.6
CC14IR           DEFB    CC14IC.7
CC15IE           DEFB    CC15IC.6
CC15IR           DEFB    CC15IC.7
CC16IE           DEFB    CC16IC.6
CC16IR           DEFB    CC16IC.7
CC17IE           DEFB    CC17IC.6
CC17IR           DEFB    CC17IC.7
CC18IE           DEFB    CC18IC.6
CC18IR           DEFB    CC18IC.7
CC19IE           DEFB    CC19IC.6
CC19IR           DEFB    CC19IC.7
CC20IE           DEFB    CC20IC.6
CC20IR           DEFB    CC20IC.7
CC21IE           DEFB    CC21IC.6
CC21IR           DEFB    CC21IC.7
CC22IE           DEFB    CC22IC.6
CC22IR           DEFB    CC22IC.7
CC23IE           DEFB    CC23IC.6
CC23IR           DEFB    CC23IC.7
CC24IE           DEFB    CC24IC.6
CC24IR           DEFB    CC24IC.7
CC25IE           DEFB    CC25IC.6
CC25IR           DEFB    CC25IC.7
CC26IE           DEFB    CC26IC.6
CC26IR           DEFB    CC26IC.7
CC27IE           DEFB    CC27IC.6
```

```
CC27IR          DEFB    CC27IC.7
CC28IE          DEFB    CC28IC.6
CC28IR          DEFB    CC28IC.7
CC29IE          DEFB    CC29IC.6
CC29IR          DEFB    CC29IC.7
CC30IE          DEFB    CC30IC.6
CC30IR          DEFB    CC30IC.7
CC31IE          DEFB    CC31IC.6
CC31IR          DEFB    CC31IC.7

ADCIE           DEFB    ADCIC.6
ADCIR           DEFB    ADCIC.7
ADEIE           DEFB    ADEIC.6
ADEIR           DEFB    ADEIC.7

T0IE            DEFB    T0IC.6
T0IR            DEFB    T0IC.7
T1IE            DEFB    T1IC.6
T1IR            DEFB    T1IC.7
T7IE            DEFB    T7IC.6
T7IR            DEFB    T7IC.7
T8IE            DEFB    T8IC.6
T8IR            DEFB    T8IC.7

ADST            DEFB    ADCON.7
ADBSY           DEFB    ADCON.8
ADWR            DEFB    ADCON.9
ADCIN           DEFB    ADCON.10
ADCRQ           DEFB    ADCON.11

ILLBUS          DEFB    TFR.0
ILLINA          DEFB    TFR.1
ILLOPA          DEFB    TFR.2
PRTFLT          DEFB    TFR.3
UNDOPC          DEFB    TFR.7
STKUF           DEFB    TFR.13
STKOF           DEFB    TFR.14
NMI             DEFB    TFR.15

WDTIN           DEFB    WDTCON.0
WDTR            DEFB    WDTCON.1

S0STP           DEFB    S0CON.3
S0REN           DEFB    S0CON.4
S0PEN           DEFB    S0CON.5
S0FEN           DEFB    S0CON.6
S0OEN           DEFB    S0CON.7
S0PE            DEFB    S0CON.8
S0FE            DEFB    S0CON.9
S0OE            DEFB    S0CON.10
S0ODD           DEFB    S0CON.12
S0BRS           DEFB    S0CON.13
S0LB            DEFB    S0CON.14
S0R             DEFB    S0CON.15

SSCHB           DEFB    SSCCON.4
SSCPH           DEFB    SSCCON.5
SSCPO           DEFB    SSCCON.6
SSCTEN          DEFB    SSCCON.8
SSCREN          DEFB    SSCCON.9
SSCPEN          DEFB    SSCCON.10
SSCBEN          DEFB    SSCCON.11
SSCBSY          DEFB    SSCCON.12
SSCMS           DEFB    SSCCON.14
SSCEN           DEFB    SSCCON.15
```

```
PTR0            DEFB    PWMCON0.0
PTR1            DEFB    PWMCON0.1
PTR2            DEFB    PWMCON0.2
PTR3            DEFB    PWMCON0.3
PTI0            DEFB    PWMCON0.4
PTI1            DEFB    PWMCON0.5
PTI2            DEFB    PWMCON0.6
PTI3            DEFB    PWMCON0.7
PIE0            DEFB    PWMCON0.8
PIE1            DEFB    PWMCON0.9
PIE2            DEFB    PWMCON0.10
PIE3            DEFB    PWMCON0.11
PIR0            DEFB    PWMCON0.12
PIR1            DEFB    PWMCON0.13
PIR2            DEFB    PWMCON0.14
PIR3            DEFB    PWMCON0.15

PEN0            DEFB    PWMCON1.0
PEN1            DEFB    PWMCON1.1
PEN2            DEFB    PWMCON1.2
PEN3            DEFB    PWMCON1.3
PM0             DEFB    PWMCON1.4
PM1             DEFB    PWMCON1.5
PM2             DEFB    PWMCON1.6
PM3             DEFB    PWMCON1.7
PB01            DEFB    PWMCON1.12
PS2             DEFB    PWMCON1.14
PS3             DEFB    PWMCON1.15

PWMIE           DEFB    PWMIC.6
PWMIR           DEFB    PWMIC.7

XP3IE           DEFB    XP3IC.6
XP3IR           DEFB    XP3IC.7
XP2IE           DEFB    XP2IC.6
XP2IR           DEFB    XP2IC.7
XP1IE           DEFB    XP1IC.6
XP1IR           DEFB    XP1IC.7
XP0IE           DEFB    XP0IC.6
XP0IR           DEFB    XP0IC.7
```

B.11 Saber to Breadboard Converter Code

On the next page starts the code for the Java Saber to Breadboard Converter tool. The files for the node are as follows.

1. SaberConverter.java
2. SaberFrame.java
3. SaberFrame_AboutBox.java

B.12 Breadboard Loads

On the next page is the file BreadBoardLoads.txt

```
//Title:      Saber to Bread Board Converter
//Version:
//Copyright:  Copyright (c) 1998
//Author:     James Geraci
//Company:    MIT LEES Lab
//Description:Saber to Bread Board Converter
package Thesis;

import com.sun.java.swing.UIManager;
import java.awt.*;
import java.io.*;
import java.util.*;
import java.text.*;
import borland.jbcl.util.*;

public class SaberConverter {
  boolean packFrame = false;

  //Construct the application

  public SaberConverter() {
    SaberFrame frame = new SaberFrame();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame)
      frame.pack();
    else
      frame.validate();
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
      frameSize.height = screenSize.height;
    if (frameSize.width > screenSize.width)
      frameSize.width = screenSize.width;
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);
    frame.setVisible(true);
  }
//Main method

  public static void main(String[] args) {
    try  {
    //  UIManager.setLookAndFeel(new com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
      //UIManager.setLookAndFeel(new com.sun.java.swing.plaf.motif.MotifLookAndFeel());
      UIManager.setLookAndFeel(new com.sun.java.swing.plaf.metal.MetalLookAndFeel());
    }
    catch (Exception e) {
    }
    new SaberConverter();

  }
}

class AlternatorRPMObject
  {
    public AlternatorRPMObject(TextField WheelDiameter, TextField DiffGearR, TextField EngAltGearR, String s)
      {
        VehicleDrivingSpeed = 0;
        TireDiameter = new Double(WheelDiameter.getText().trim()).doubleValue();
        DifferentialGearRatio = new Double(DiffGearR.getText().trim()).doubleValue();
        TransmissionGearRatio = 0;
        EngineAlternatorGearRatio = new Double(EngAltGearR.getText().trim()).doubleValue();
        AlternatorShaftSpeed = 0;
        TimeOfEvent = 0;
        try
          {
            GenerateAlternatorShaftSpeed(s);
          }
        catch(IOException rt)
          {
            System.exit(1);
          }
      }

  public void GenerateAlternatorShaftSpeed(String s) throws IOException
      {
        String Time;
        String Speed;
        String Gear;

        StringTokenizer token = new StringTokenizer(s, " \t\n\r");
        if(token.hasMoreTokens())
        {
        Time = token.nextToken();
        Speed = token.nextToken();
        Gear = token.nextToken();

        int TimeLength = Time.length();
        int SpeedLength = Speed.length();
        int GearLength = Gear.length();
```

```
double TimeDataDouble = new Double(Time.substring(0, TimeLength)).doubleValue();
double SpeedDataDouble = new Double(Speed.substring(0, SpeedLength)).doubleValue();
double GearDataDouble = new Double(Gear.substring(0, GearLength)).doubleValue();

VehicleDrivingSpeed = SpeedDataDouble;
int TimeDataInteger = (int) (TimeDataDouble);
int SpeedDataInteger = (int) (SpeedDataDouble);
int GearDataInteger = (int) (GearDataDouble );

if(GearDataInteger == 0  && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 0 ;
  }
else if(GearDataInteger == 1 && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 3.071;
  }
else if(GearDataInteger == 2 && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 1.773;
  }
else if(GearDataInteger == 3 && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 1.194;
  }
else if(GearDataInteger == 4 && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 0.868;
  }
else if(GearDataInteger == 5 && SpeedDataInteger != -1)
  {
    TransmissionGearRatio = 0.700;
  }
else
  {
    TransmissionGearRatio = 0;
  }

if (GearDataInteger == 0 && SpeedDataInteger != -1)
  {
    AlternatorShaftSpeed = 600;
  }
else if (SpeedDataInteger == -1)
  {
    AlternatorShaftSpeed = 0;
  }
```

```
        else
          {
          AlternatorShaftSpeed = (TransmissionGearRatio*((10.0/36.0)*(60.0)/(TireDiameter*(Math.PI)))*DifferentialGearRatio*V
ehicleDrivingSpeed);
          if((AlternatorShaftSpeed == 0) || (AlternatorShaftSpeed < 600))
            {
              AlternatorShaftSpeed = 600;
            }
          }
        Time = Integer.toString(TimeDataInteger);
        Speed = Integer.toString(SpeedDataInteger);
        Gear = Integer.toString(GearDataInteger);

       AlternatorSpeed = Integer.toString((int) AlternatorShaftSpeed);
       }
      }

      public String getAlternatorShaftSpeed()
      {
        //return VehicleDrivingSpeed;
        return AlternatorSpeed;
      }
        public double getAlternatorShaftSpeed2()
      {
        //return VehicleDrivingSpeed;
        return AlternatorShaftSpeed;
      }

    private String AlternatorSpeed;
    private double VehicleDrivingSpeed;
    private double TireDiameter;
    private double DifferentialGearRatio;
    private double TransmissionGearRatio;
    private double EngineAlternatorGearRatio;
    private double AlternatorShaftSpeed;
    private double TimeOfEvent;

  }


class CANEventObject
  {
    CANEventObject()
      {

      }
```

```
public void CANEventObjectFileHandler(String s, int xx)
  {
    //  System.out.println(xx);
      xx++;
try{
    BufferedReader SCSFileIn = new BufferedReader(new FileReader(s), 20000);
    String s2;
    while((s2 = SCSFileIn.readLine() )!= null)
      {
       // s2.trim();
        workwithCANString(s2);
      }
        trimVectors();
        SCSFileIn.close();

      /* This Section of Code Deals with Extension Files */
    int x = CANEventGenerators.size() - 1;
    int y = 0;
    while (y < x)
      {
        if(((CANObjectClass)(CANEventGenerators.elementAt(y))).returnDoExtensionFilesExist())
          {

            String s3 = ((CANObjectClass)(CANEventGenerators.elementAt(y))).returnNameOfExtensionFile();
               ((CANObjectClass)(CANEventGenerators.elementAt(y))).setDoExtensionFilesExist(false);
            CANEventObjectFileHandler(s3, xx);
          }
        y++;
      }
    }

  catch(IOException ex)
    {
      System.exit(1);
    }

  }


public void workwithCANString(String inputstring)
    {
      StringTokenizer s2 = new StringTokenizer(inputstring, "\n\r");
      if(s2.hasMoreElements())
      {
      String s = s2.nextToken();
```

```
        int indexofdecimalpoint = 0;
        int indexoflastfrontslash = 0;
        int indexofopenbrace = 0;
        int indexofclosebrace = 0;
        String NameOfCANEvent;
        String OnandOffTimes;
        int CANEventVectorSize = 0;
        int CANCounterSize = 0;
        int counter = 0;
        int counter2 = 0;

        boolean AlreadyExists = false;
        String AppendFileName;
        s.trim();
         if(!(s.startsWith("#")))
          {
            if(s.startsWith("alter"))
            {
            indexofdecimalpoint = s.indexOf(".");
            indexoflastfrontslash = s.lastIndexOf("/");
            indexofopenbrace = s.indexOf("[");
            indexofclosebrace = s.indexOf("]");
            NameOfCANEvent = s.substring((indexofdecimalpoint + 1), indexoflastfrontslash);
            CANCounterSize = CANEventGenerators.size();

            while(counter2 < CANCounterSize)
              {
                if (NameOfCANEvent.equals(((CANObjectClass) (CANEventGenerators.elementAt(counter2))).returnCANEventName()))
                {
                  CANEventVectorSize = counter2;
                  counter2 = CANCounterSize + 1;
                  AlreadyExists = true;
                }
                counter2++;
               }
            if(!AlreadyExists)
              {
              CANEventGenerators.addElement(new CANObjectClass(NameOfCANEvent));
              if(CANEventGenerators.size() != 0)
              CANEventVectorSize = CANEventGenerators.size() - 1;
              }

            if(indexofclosebrace != -1)
              {
               ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setOnandOffTimes(s.substring(indexofopen
brace + 1, indexofclosebrace));
```

```
              }
            else
              {
               ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setOnandOffTimes(s.substring(indexofopen
brace + 1));
              }
            }
            if(s.startsWith("(") || s.startsWith(","))
              {
                if(indexofclosebrace != -1)
                  {
                  ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setOnandOffTimes(s.substring(indexofo
penbrace + 1, indexofclosebrace));
                  }
                else
                  {
                  ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setOnandOffTimes(s.substring(indexofo
penbrace + 1));
                  }
              }
        }
        else if(s.startsWith("#") && s.endsWith(".scs"))
          {
           StringTokenizer token = new StringTokenizer(s, " \t\n\r");
           int TokenCount = token.countTokens();
           int x = 1;
              while(x < TokenCount)
                  {
                  if(token.hasMoreElements())
                    {
                    token.nextToken();
                    }
                    x++;
                  }
           ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setNameOfExtensionFile(token.nextToken());
          ((CANObjectClass) (CANEventGenerators.elementAt(CANEventVectorSize))).setDoExtensionFilesExist(true);
          }
     }
 }
    public void trimVectors()
      {
       CANEventGenerators.trimToSize();
       }
    public int returnCANEventListSize()
      {
    //     return CANEventGenerators.size();
```

```
  //   // return BreadBoardCANLoads.size();
 //    return ValidCANEventGenerators.size();
//    return ProgrammableLoadPowerDemanded.size();
      return FinalCANList.size();
      }

  public String returnProgrammableLoad(int x)
    {
     return ((String) ProgrammableLoadPowerDemanded.elementAt(x));
    }

    public String returnCANString(int x)
      {
      //  return ((CANObjectClass)(CANEventGenerators.elementAt(x))).returnOnandOffTimes();
   //   return ((String) (BreadBoardCANLoads.elementAt(x)));

    return (((CANMessageClass)(FinalCANList.elementAt(x))).returnCANMessage());
   //   return ((CANObjectClass)(ValidCANEventGenerators.elementAt(x))).returnCANEventName();
      }

public int returnCANEventTime(int x)
  {
    return (((CANMessageClass)(FinalCANList.elementAt(x))).returntime());
  }



public void ReadinBreadBoardCANLoads(String inputfile)
      {

       try{
        BufferedReader filein = new BufferedReader(new FileReader(inputfile.trim()));
        String s;

//  Here is where the actual load list is read in.
        while((s = filein.readLine()) != null)
          {
            handleCANString(s);
          }
          filein.close();
        }
       catch(IOException ex)
        {
        System.exit(1);
        }
      }
 private void handleCANString(String x)
  {
    StringTokenizer token = new StringTokenizer(x, " \t\n\r");
    if(token.hasMoreElements())
      {
      String CanName1 = (String) token.nextElement();
      String Message_ID = (String) token.nextElement();
      // Here is where the load is actually added to the list
      BreadBoardCANLoads.addElement(new CANObjectClass(CanName1, Message_ID));

      }
  }

public void ConfirmBreadBoardCompatability()
    {
      while(CANEventGenerators.size() > 0)
        {
      //  System.out.println(CANEventGenerators.size() + "\t" + ValidCANEventGenerators.size() + "\t" + NotValidCANEventGener
ators.size());
        boolean istrue = false;
        int BreadBoardLoads = BreadBoardCANLoads.size() - 1;
        int count = 0;
        int holder = 0;
        while(count < BreadBoardLoads)
          {
            if((((CANObjectClass)(BreadBoardCANLoads.elementAt(count))).returnCANEventName()).equals(((CANObjectClass) (CANEv
entGenerators.elementAt(0))).returnCANEventName()))
              {
                istrue = true;
                holder = count;
              }
            count++;
          }
          if(istrue)
            {
            ValidCANEventGenerators.addElement((CANObjectClass) (CANEventGenerators.elementAt(0)));
            ((CANObjectClass) ValidCANEventGenerators.lastElement()).setMessageID(((CANObjectClass)(BreadBoardCANLoads.elemen
tAt(holder))).returnMessageID());
            CANEventGenerators.removeElementAt(0);
             }
            else
              {
                NotValidCANEventGenerators.addElement((CANObjectClass) (CANEventGenerators.elementAt(0)));
                CANEventGenerators.removeElementAt(0);
              }
        }
        ValidCANEventGenerators.trimToSize();
    }

  public void GenerateEMValvePowerDemand(TextField HigherVoltage, TextField LowerVoltage, TextField PowerAvailable, Vector Al
ternatorSpeedVector)
    {
     double HigherBusVoltage = new Double(HigherVoltage.getText().trim()).doubleValue();
     double LowerBusVoltage = new Double(LowerVoltage.getText().trim()).doubleValue();
     double ProgrammablePowerAvailable = 1800; //new Double(PowerAvailable.getText().trim()).doubleValue();
     double IdleRPMSpeed  = 600;  // From Irene Quo’s Master Thesis page 85 of motor not alternator
     double HighSpeedRPMSpeed = 2000; // From Irene Quo’s Master Thesis page 85
     double MaxCurrent = ProgrammablePowerAvailable / HigherBusVoltage;
     double MinCurrent = MaxCurrent / 5;
     double SizeofAlternatorSpeedVector = AlternatorSpeedVector.size();
     double Slope = (MaxCurrent - MinCurrent)/(HighSpeedRPMSpeed - IdleRPMSpeed);
  //   System.out.println("This is Computing the EMValve Power Demanded");
  System.out.println("Slope = " + Slope);
     int counter = 0;

      while(counter < SizeofAlternatorSpeedVector)
        {
          if((Double.valueOf((((AlternatorRPMObject) (AlternatorSpeedVector.elementAt(counter))).getAlternatorShaftSpeed())).
doubleValue()) < IdleRPMSpeed)
             {
              ProgrammableLoadPowerDemanded.addElement(Double.toString(0));
              }
          else if((Double.valueOf((((AlternatorRPMObject) (AlternatorSpeedVector.elementAt(counter))).getAlternatorShaftSpeed
())).doubleValue()) >= HighSpeedRPMSpeed)
              {
                ProgrammableLoadPowerDemanded.addElement(Double.toString(MaxCurrent));
              }
          else
            {
              int Current = (int) (Slope*((Double.valueOf((((AlternatorRPMObject) (AlternatorSpeedVector.elementAt(counter)))
.getAlternatorShaftSpeed())).doubleValue())) - 6.425);
              if (Current <= MaxCurrent)
                {
                  ProgrammableLoadPowerDemanded.addElement(Double.toString(Current));
                }
              else
                {
                  ProgrammableLoadPowerDemanded.addElement(Double.toString(MaxCurrent));
                }
            }
         counter++;
        }
   // System.out.println("EMValve stuff computed");
    ProgrammableLoadPowerDemanded.trimToSize();
    }

    public void CreateCANMessages()
    {
      int y = 0;
      while(y < ValidCANEventGenerators.size())
        {
        parseCANString(((CANObjectClass) (ValidCANEventGenerators.elementAt(y))).returnOnandOffTimes(), ((CANObjectClass) Val
idCANEventGenerators.elementAt(y)));
   //     System.out.println(((CANObjectClass) (ValidCANEventGenerators.elementAt(y))).returnOnandOffTimes());
        y++;
        }
     }

     private void parseCANString(String s, CANObjectClass sClass)
      {
      StringTokenizer token = new StringTokenizer(s, "(");
      // Test to see if Tokens exist

            while(token.hasMoreElements())
              {
              String snext = token.nextToken();
              SemiFinalCANList.addElement(new CANMessageClass(snext, sClass));
               }
      }

     public void removeCANString(int x)
      {
        FinalCANList.removeElementAt(x);
      }

     public void removeT0offMessages()
     {
      while(SemiFinalCANList.size() != 0)
        {
        if((((CANMessageClass) SemiFinalCANList.elementAt(0)).returntime()) == 0)
          {
            if(!(((CANMessageClass) (SemiFinalCANList.elementAt(0))).returnTurnOn()))
              {
                SemiFinalCANList.removeElementAt(0);
              }
            }
        else
```

```
          {
            FinalCANList.addElement((CANMessageClass) (SemiFinalCANList.elementAt(0)));
            SemiFinalCANList.removeElementAt(0);
          }
        }
     }

      private String CANString;
      private Vector FinalCANList = new Vector(100, 20);
      private Vector SemiFinalCANList = new Vector(100, 20);
      private Vector BreadBoardCANLoads = new Vector(11);
      private Vector CANEventGenerators = new Vector(100, 20);
      private Vector ValidCANEventGenerators = new Vector(11);
      private Vector NotValidCANEventGenerators = new Vector(11);
      private Vector ProgrammableLoadPowerDemanded = new Vector(30000, 500);
    //  private Vector ProgrammableLoadLoads = new Vector(11);
 }

class CANMessageClass
  {
    CANMessageClass(String s, CANObjectClass CANObject)
      {
      StringTokenizer token = new StringTokenizer(s, ",");
        if(token.hasMoreElements())
          {
            time =  (int) Integer.parseInt(token.nextToken());

          String e = token.nextToken();
        StringTokenizer token2 = new StringTokenizer(e, ")");
//  Compute most of the checksum
          byte[] buffer = new byte[4];
                buffer = (CANObject.returnMessageID()).getBytes();
                int idvalue = ConvertFromText(buffer);

                int checksum = 0;
                checksum = 3 + 8 + idvalue;  // not done with the checksum just yet
        //  Determine if you are turning the switch on or off
          e = (token2.nextToken()).trim();

         if(e.equals("2") || e.equals("3") || e.equals("4"))
              {
                checksum = checksum + 1;
                String HexString = ConvertToHex(checksum);
                CANMessage = "A00308"+ (CANObject.returnMessageID()) + "0000010000000000" + HexString + "0A"; // Turn the 42V
olt Heater On
                TurnOn = true;
              }
              else if (e.equals("1"))
              {
                checksum = checksum + 8;
                String HexString = ConvertToHex(checksum);
                CANMessage = "A00308" + (CANObject.returnMessageID()) + "0008000000000000" + HexString + "0A"; // Turn the 42
Volt Heater Off
                TurnOn = false;
              }
           }
        }

    private int ConvertFromText(byte[] byter)
      {
        int sum = 0;
        int int0 = (int) byter[0];
        int int2 = (int) byter[2];
        int int3 = (int) byter[3];
        if(int0 >= 48 && int0 <= 57)
          {
            int0 = int0 - 48;
          }
        else if(int0 >= 65 && int0 <= 70)
          {
            int0 = int0 - 55;
          }
        if(int2 >= 48 && int2 <= 57)
          {
            int2 = int2 - 48;
          }
        else if(int2 >= 65 && int2 <= 70)
          {
            int2 = int2 - 55;
          }
        if(int3 >= 48 && int3 <= 57)
          {
            int3 = int3 - 48;
          }
        else if(int3 >= 65 && int3 <= 70)
          {
            int3 = int3 - 55;
          }

        int0 = int0 * 16;
        int2 = int2 * 16;
        sum = int0 + int2 + int3;
```

```
   return sum;
  }

private String ConvertToHex(int checksum)
  {
   char[] chararray = new char[4];
   int lowestnibble = checksum & 15;
   int secondnibble = checksum & 240;
   int thirdnibble = checksum &  3840;
   int topnibble = checksum & 61440;
   secondnibble = secondnibble >>> 4;
   thirdnibble = thirdnibble >>>  8;
   topnibble = topnibble >>> 12;
   chararray[0] = FindLetter(topnibble);
   chararray[1] = FindLetter(thirdnibble);
   chararray[2] = FindLetter(secondnibble);
   chararray[3] = FindLetter(lowestnibble);

   String sammy = new String(chararray);
   return sammy;
  }

private char FindLetter(int x)
  {
    char y = '0';
    if(x >= 10)
      {
      y = (char) (x + 55);
      }
    else if(x <= 9)
      {
        y = (char) (x + 48);
      }
    return y;
    }


public boolean returnTurnOn()
  {
    return TurnOn;
  }

public int returntime()
  {
    return time;
  }
```

```
    public String returnCANMessage()
      {
        return CANMessage;
      }
     private int time;
     private boolean TurnOn;
     private String CANMessage;
    }


class CANObjectClass
  {
    CANObjectClass(String CName, String M_ID)
      {
        CANEventName = CName;
        Message_ID = M_ID;
      }

    CANObjectClass(String s)
      {
       CANEventName = s;
      }

    public void setOnandOffTimes(String s)
      {
        String s2 = s.trim();
        if(!append)
          {
            OnandOffTimes = s;
            append = true;
          }
        else
          {
            OnandOffTimes = OnandOffTimes + s;
          }
      }

    public void appendOnandOffTimes(String s)
      {
        OnandOffTimes = OnandOffTimes + s;
      }

    public void setNameOfExtensionFile(String s)
      {
        NameOfExtensionFile = s;
    }

  public String returnNameOfExtensionFile()
    {
      return NameOfExtensionFile;
      }


  public String returnOnandOffTimes()
    {
      return OnandOffTimes;
    }

  public String returnCANEventName()
    {
      return CANEventName;
    }

  public boolean returnDoExtensionFilesExist()
    {
      return DoExtensionFilesExist;
    }

  public void setDoExtensionFilesExist(boolean t)
    {
      DoExtensionFilesExist = t;
    }

  public void setMessageID(String s)
    {
      Message_ID = s;
    }

  public String returnMessageID()
    {
      return Message_ID;
    }

  private boolean append = false;
  private String CANEventName;
  private String OnandOffTimes;
  private String NameOfExtensionFile = "No Extension Files";
  private String Message_ID;
  private boolean DoExtensionFilesExist = false;
  private Vector OffTimes = new Vector(20);
}
```

```
//Title:       Saber to Bread Board Converter
//Version:
//Copyright:   Copyright (c) 1998
//Author:      James Geraci
//Company:     MIT LEES Lab
//Description:Saber to Bread Board Converter
package Thesis;

import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import borland.jbcl.control.BevelPanel;
import borland.jbcl.control.ImageControl;

public class SaberFrame_AboutBox extends Dialog implements ActionListener {

  BevelPanel panel1 = new BevelPanel();
  BevelPanel panel2 = new BevelPanel();
  BevelPanel insetsPanel1 = new BevelPanel();
  BevelPanel insetsPanel2 = new BevelPanel();
  BevelPanel insetsPanel3 = new BevelPanel();
  JButton button1 = new JButton();
  ImageControl imageControl1 = new ImageControl();
  JLabel label1 = new JLabel();
  JLabel label2 = new JLabel();
  JLabel label3 = new JLabel();
  JLabel label4 = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  BorderLayout borderLayout2 = new BorderLayout();
  FlowLayout flowLayout1 = new FlowLayout();
  FlowLayout flowLayout2 = new FlowLayout();
  GridLayout gridLayout1 = new GridLayout();
  String product = "Saber to Bread Board Converter";
  String version = "";
  String copyright = "Copyright (c) 1998";
  String comments = "Saber to Bread Board Converter";

  public SaberFrame_AboutBox(Frame parent) {
    super(parent);
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try  {
      jbInit();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    pack();
  }

  private void jbInit() throws Exception  {
    this.setTitle("About");
    setResizable(false);
    panel1.setLayout(borderLayout1);
    panel2.setLayout(borderLayout2);
    insetsPanel1.setLayout(flowLayout1);
    insetsPanel1.setBevelInner(BevelPanel.FLAT);
    insetsPanel2.setLayout(flowLayout1);
    insetsPanel2.setMargins(new Insets(10, 10, 10, 10));
    insetsPanel2.setBevelInner(BevelPanel.FLAT);
    gridLayout1.setRows(4);
    gridLayout1.setColumns(1);
    label1.setText(product);
    label2.setText(version);
    label3.setText(copyright);
    label4.setText(comments);
    insetsPanel3.setLayout(gridLayout1);
    insetsPanel3.setMargins(new Insets(10, 60, 10, 10));
    insetsPanel3.setBevelInner(BevelPanel.FLAT);
    button1.setText("OK");
    button1.addActionListener(this);
    imageControl1.setImageName("");
    insetsPanel2.add(imageControl1, null);
    panel2.add(insetsPanel2, BorderLayout.WEST);
    this.add(panel1, null);
    insetsPanel3.add(label1, null);
    insetsPanel3.add(label2, null);
    insetsPanel3.add(label3, null);
    insetsPanel3.add(label4, null);
    panel2.add(insetsPanel3, BorderLayout.CENTER);
    insetsPanel1.add(button1, null);
    panel1.add(insetsPanel1, BorderLayout.SOUTH);
    panel1.add(panel2, BorderLayout.NORTH);
  }

  protected void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      cancel();
    }
    super.processWindowEvent(e);
  }
  void cancel() {
    dispose();
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
      cancel();
    }
  }
}
```

```
//Title:      Saber to Bread Board Converter
//Version:
//Copyright:  Copyright (c) 1998
//Author:     James Geraci
//Company:    MIT LEES Lab
//Description:Saber to Bread Board Converter
package Thesis;

import java.awt.*;
import java.awt.event.*;
import borland.jbcl.control.*;
import borland.jbcl.layout.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class SaberFrame extends DecoratedFrame {

  //Construct the frame
  BorderLayout borderLayout1 = new BorderLayout();
  XYLayout xYLayout2 = new XYLayout();
  BevelPanel bevelPanel1 = new BevelPanel();
  MenuBar menuBar1 = new MenuBar();
  Menu menuFile = new Menu();
  MenuItem menuFileExit = new MenuItem();
  Menu menuHelp = new Menu();
  MenuItem menuHelpAbout = new MenuItem();
  ButtonBar toolBar = new ButtonBar();
  StatusBar statusBar = new StatusBar();
  TextField textField1 = new TextField();
  Button button1 = new Button();
  TextField textField2 = new TextField();
  TextField textField3 = new TextField();
  Label label1 = new Label();
  Label label2 = new Label();
  TextField textField4 = new TextField();
  Label label3 = new Label();
  TextField textField5 = new TextField();
  TextField textField6 = new TextField();
  TextField textField7 = new TextField();
  TextField textField8 = new TextField();
  TextField textField9 = new TextField();
  Label label4 = new Label();
  Label label5 = new Label();
  TextField textField10 = new TextField();
  Label label6 = new Label();
  TextField textField11 = new TextField();
  Label label8 = new Label();
  Label label9 = new Label();

  public SaberFrame() {
    try  {
      jbInit();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
//Component initialization


  private void jbInit() throws Exception  {
    this.setLayout(borderLayout1);
    this.setSize(new Dimension(466, 358));
    this.setTitle("Saber to BreadBoard Converter Program");
    menuFile.setLabel("File");
    menuFileExit.setLabel("Exit");
    menuFileExit.addActionListener(new ActionListener()  {
      public void actionPerformed(ActionEvent e) {
        fileExit_actionPerformed(e);
      }
    });
    menuHelp.setLabel("Help");
    menuHelpAbout.setLabel("About");
    menuHelpAbout.addActionListener(new ActionListener()  {
      public void actionPerformed(ActionEvent e) {
        helpAbout_actionPerformed(e);
      }
    });
    toolBar.setButtonType(ButtonBar.IMAGE_ONLY);
    toolBar.setLabels(new String[] {"File", "Close", "Help"});
    textField1.setText("ece15_city.dat");
    button1.setLabel("GenerateFile");
    textField2.setText("0.594");
    textField3.setText("4.0");
    label1.setText("Tire Diameter :");
    label2.setText("Differential Gear Ratio :");
    textField4.setText("3.0");
    label3.setText("Engine-Alternator Gear Ratio :");
    textField5.setText("winter_worst_ece15.scs");
    textField6.setText("textField6");
    textField7.setText("BreadBoardCANLoads.txt");
```

```
textField8.setText("40");
textField9.setText("14");
label4.setText("Higher Voltage Bus Voltage :");
label5.setText("Lower Voltage Bus Voltage :");
textField10.setText("1800");
label6.setText("Programmable Load Wattage :");
textField11.setText("BBInputFile.txt");
label8.setText("e :");
label9.setText("Output FileName :");

button1.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    button1_actionPerformed(e);
  }
});
toolBar.setImageBase("image");
toolBar.setImageNames(new String[] {"openFile.gif", "closeFile.gif", "help.gif"});
bevelPanel1.setLayout(xYLayout2);
menuFile.add(menuFileExit);
menuHelp.add(menuHelpAbout);
menuBar1.add(menuFile);
menuBar1.add(menuHelp);
this.setMenuBar(menuBar1);
this.add(toolBar, BorderLayout.NORTH);
this.add(statusBar, BorderLayout.SOUTH);
this.add(bevelPanel1, BorderLayout.WEST);
bevelPanel1.add(textField1, new XYConstraints(7, 20, 201, -1));
bevelPanel1.add(button1, new XYConstraints(349, 12, 99, 35));
bevelPanel1.add(textField2, new XYConstraints(388, 52, 60, 21));
bevelPanel1.add(textField3, new XYConstraints(388, 79, 60, 21));
bevelPanel1.add(label1, new XYConstraints(292, 52, -1, -1));
bevelPanel1.add(label2, new XYConstraints(249, 79, -1, -1));
bevelPanel1.add(textField4, new XYConstraints(388, 104, 60, 21));
bevelPanel1.add(label3, new XYConstraints(211, 102, -1, -1));
bevelPanel1.add(textField5, new XYConstraints(7, 52, 201, -1));
bevelPanel1.add(textField6, new XYConstraints(1, 244, 172, 34));
bevelPanel1.add(textField7, new XYConstraints(7, 79, 201, -1));
bevelPanel1.add(textField8, new XYConstraints(388, 129, 60, 21));
bevelPanel1.add(textField9, new XYConstraints(388, 157, 60, 21));
bevelPanel1.add(label4, new XYConstraints(220, 129, -1, -1));
bevelPanel1.add(label5, new XYConstraints(224, 156, -1, -1));
bevelPanel1.add(textField10, new XYConstraints(388, 185, 60, 21));
bevelPanel1.add(label6, new XYConstraints(208, 183, 155, 25));
bevelPanel1.add(textField11, new XYConstraints(7, 200, 161, -1));
bevelPanel1.add(label8, new XYConstraints(364, 183, 23, -1));
bevelPanel1.add(label9, new XYConstraints(7, 176, -1, -1));
```

```
  }
//File | Exit action performed

  public void fileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }
//Help | About action performed

  public void helpAbout_actionPerformed(ActionEvent e) {
    SaberFrame_AboutBox dlg = new SaberFrame_AboutBox(this);
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height - dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.show();
  }

void button1_actionPerformed(ActionEvent e)
  {
    textField6.setText("I have Started");
    Vector AlternatorInfoVector = new Vector(50000, 500);
    CANEventObject CANCollectorObject = new CANEventObject();

    /* Generate the Alternator RPM by reading in a *.dat file and then putting all of
    the data into a vecotr that contains AlternatorRPMObjects
      */

try{
    BufferedReader filein = new BufferedReader(new FileReader(textField1.getText()), 2000000);
    String s;
    while((s = filein.readLine()) != null)
    {
      s.trim();
      AlternatorInfoVector.addElement(new AlternatorRPMObject(textField2, textField3, textField4, s));
    }
     AlternatorInfoVector.trimToSize();
     filein.close();
  }
  catch(IOException ex)
  {
    System.exit(1);
  }

 /* Now Read in the SCS File and put the data into a Vector which contains
 objects of type CANEventObject*/
```

```
   System.out.println("You are Starting the CANEventObjectFileHandler");
  CANCollectorObject.CANEventObjectFileHandler(textField5.getText(), 0);
    System.out.println("You have completed the CANEventObjectFileHandler");
    System.out.println("You are now starting the ReadinBreadBoardCANLoads");
  /* The following function reads in a list of all known BreadBoardCANLoads*/

  CANCollectorObject.ReadinBreadBoardCANLoads(textField7.getText());

  /* The following function checks to see if the loads used in the Saber Simulation are
    Available on the CAN bus */
    System.out.println("You are now starting the ConfirmBreadBoardCompatability");
  CANCollectorObject.ConfirmBreadBoardCompatability();

  /* The following function generates the serial messages which are to be used
    to activate the events on the CAN BUS it also puts them togther with their
    appropriate Alternator RPM Object*/
    System.out.println("You are now starting the CreateCANMessages");
  CANCollectorObject.CreateCANMessages();
  /*  The following function removes all the the Turn off Commands at t=0 */
    System.out.println("You are now removing the excess turn off commands");
    CANCollectorObject.removeT0offMessages();

  /* The following function generates the appropriate ElectroMechanical Valve Power Demand
    For a given Alternator Speed*/
  System.out.println("You are now starting the GenerateEMValvePowerDemand");
  CANCollectorObject.GenerateEMValvePowerDemand(textField8, textField9, textField10, AlternatorInfoVector);


  //     Generate the Output File
  System.out.println("Now Writing the output file");
 // int RunCounter = 0;
 // double NumberOfRunsDouble = new Double(textField13.getText()).doubleValue();
 // int NumberOfRuns = (int) NumberOfRunsDouble;

try{

    PrintWriter out = new PrintWriter(new FileOutputStream((textField11.getText()).trim()));

  //   while(RunCounter < NumberOfRuns)
 // {
    int startupbuffer = 0;
    while(startupbuffer < 60)
      {
        out.println("//");
        startupbuffer++;
      }
    int x = 0;
    //int CANEventListSize = CANCollectorObject.returnCANEventListSize();
    int z = AlternatorInfoVector.size() - 1;
    int peter = 0;
    double previousProgrammableLoad = 0;
    int previousRPM = -1;
    int fourthpoint = 1;
    int testRPM = 0;
    while(x < (z - 1))
     {
   //   System.out.println(x);
     out.print("!" + x);
   //  out.print(x);
     if(previousRPM != ((int) Integer.parseInt(((AlternatorRPMObject) (AlternatorInfoVector.elementAt(x))).getAlternatorShaft
Speed())))
        {
         //    testRPM = ((int) Integer.parseInt(((AlternatorRPMObject) (AlternatorInfoVector.elementAt(x))).getAlternatorSha
ftSpeed()));
         //   testRPM = testRPM + 1;
        // out.print("\t" + "?" + previousRPM);
         // System.out.println(previousRPM + " " + ((int) Integer.parseInt(((AlternatorRPMObject) (AlternatorInfoVector.eleme
ntAt(x))).getAlternatorShaftSpeed())));
          out.print("\t" + "?" + ((AlternatorRPMObject) (AlternatorInfoVector.elementAt(x))).getAlternatorShaftSpeed());
               previousRPM = ((int) Integer.parseInt(((AlternatorRPMObject) (AlternatorInfoVector.elementAt(x))).getAlternato
rShaftSpeed()));


     }
     //out.print("\t");
     peter = 0;


    while(peter < (CANCollectorObject.returnCANEventListSize()))
      {
        if((CANCollectorObject.returnCANEventTime(peter)) == x)
          {
           out.print("\t" + "#" + CANCollectorObject.returnCANString(peter));
          // CANCollectorObject.removeCANString(peter);
          }
          peter++;
      }
       if(previousProgrammableLoad != Double.valueOf(CANCollectorObject.returnProgrammableLoad(x)).doubleValue())
          {
            out.print("\t" + "^42+");
            out.print(CANCollectorObject.returnProgrammableLoad(x));
          }
     previousProgrammableLoad = Double.valueOf(CANCollectorObject.returnProgrammableLoad(x)).doubleValue();
     if(fourthpoint == 1)
      {

      // These are the data collection CAN Calls.
      out.print("\t" + "#A0030000050000000000000000000080A");

      }
      else if(fourthpoint ==2)
      {

      out.print("\t" + "#A0030000BA0000000000000000000BD0A");
      }

     /* out.print("\t" + "#A00300000700000000000000000000A0A");
      out.print("\t" + "#A0030000080000000000000000000B0A");
       }*/
    else if(fourthpoint == 3)
      {
       out.print("\t" + "#A00300000F0000000000000000000120A");
       }
     else if(fourthpoint ==4)
      {
      out.print("\t" + "#A0030000090000000000000000000C0A");
      }
      else if(fourthpoint ==5)
      {
      out.print("\t" + "#A0030000060000000000000000000090A");
      fourthpoint = 0;
      }

      // out.print("\t" + "?" + ((AlternatorRPMObject) (AlternatorInfoVector.elementAt(x))).getAlternatorShaftSpeed());
      out.println("\t" + "//");
      fourthpoint++;
     x++;
   }
// RunCounter++;
// }
     out.close();
}

catch(IOException ex)
{
  System.exit(1);
  }
  textField6.setText("I'm Done");
```

```
    System.out.println("I’m Done");
  }
}
```

```
sdr_locks 0001
sdr_driver  2001
sdr_turn 8001
sdr_brakes C001
sdr_abs_tc E002
sdr_defog  0016
sdr_heater 8003
sdr_rear_seat_htrs 0019
sdr_emissions 0004
sdr_windshield  4004
sdr_seat_htrs  0003
```

Bibliography

[1] Richard A. Perez, *The Complete Battery Book*, p. 134, Tab Books Inc, Blue Ridge Summit, PA 17214, first edition, 1985c1985, ISBN 0-83-6-0757-9.

[2] E.E. Morton Arendt, *Storage Batteries Theory, Manufacture, Care, and Application*, p. 22, D. Van Nostrand Company, Inc., Eight Warren Street, New York, 1928c1928.

[3] *ACDelco 1999 Batteries Catalog 7A-100.*

[4] John Kassakian, "Automotive electrical systems circa 2005," *IEEE Spectrum*, 1997, http://auto.mit.edu/Consortia.nsf/ArticleViews.

[5] Wolfhard Lawrenz, *CAN System Engineering: From Theory to Practical Applications*, Springer Verlag, 1997c1997, ISBN 0387949399.

[6] Siemens AG, *C167 Derivatives, 16-Bit CMOS Single-Chip Microcontroller*, 2.0 edition, Apr.-May 1996, Section 8.

[7] Siemens AG, *C167 Derivatives, 16-Bit CMOS Single-Chip Microcontroller*, 2.0 edition, Apr.-May 1996, Section 16.

[8] Irene Kuo, "A methodology for sizing components in a dual-voltage automotive electrical system," Tech. Rep., Massachusetts Institute of Technology, 1999.

[9] James K. Roberge, *Operational Amplifiers, Theory and Practice*, p. 458, John Wiley & Sons, Inc., New York, 1975c1975, ISBN 0-471-72585-4.