School of Engineering Science
Burnaby, BC V5A 1S6
nvision-tech@sfu.ca

October 27, 2003

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 340 Design Specification for an Automotive Diagnostic Tool

Dear Dr. Rawicz:

The attached document, *Design Specification for an Automotive Diagnostic System*, outlines our design project for ENSC 340 (Engineering Project). We are in the process of designing a system that will interface with existing diagnostic ports on late-model automobiles, thereby providing hobby mechanics and sport enthusiasts with inexpensive access to detailed information from the car's computer.

This design specification outlines the implementation of the various capabilities of the *Automotive Diagnostic Tool* stated in our functional specification. Included in this document is a complete detailing of the hardware, firmware, software and system interfaces required for the proof-of-concept prototype. Also included is the system test plan. The proof-of-concept device will be delivered by mid-December, 2003.

NVision Technologies is a group of four talented, enthusiastic and creative third-year engineering students: Jozsef Dudas, Seema Jaffer, Deanna Lee, and Byron Thom. Should you have any questions or concerns, please feel free to contact me via telephone at 604-773-9712 or email at nvision-tech@sfu.ca.

Sincerely,

Joe Dudas

Jozsef Dudas
President and CEO
NVision Technologies

Enclosure: Design Specification for an Automotive Diagnostic System

Design Specification for an

Automotive Diagnostic Tool

NVision Technologies:	Jozsef Dudas Seema Jaffer Deanna Lee Byron Thom
Contact Person:	Jozsef Dudas NVision-Tech@sfu.ca
Submitted to:	Dr. Andrew Rawicz – ENSC 340 Steve Whitmore – ENSC 305 School of Engineering Science Simon Fraser University

Issue Date:	October 27, 2003
Revision:	0

Executive Summary

The constantly increasing complexity of recent automobiles requires hobby mechanics and sports car enthusiasts to purchase expensive electronic tools to appropriately enjoy their hobby. With the automotive diagnostic tool, users can communicate directly with the computer in their car, allowing them to retrieve real-time sensor data and malfunction codes, as well as modify operating parameters in the computer at an affordable price. Enthusiasts can then diagnose problems and improve performance at the click of an economical button!

Development of the product will occur in multiple stages. The first stage, to be completed by mid-December, 2003, will be a proof-of-concept device including the following features:

- Full communications link with the car's onboard computer in late-model GM vehicles, via the SAE J1850 VPW protocol.
- User Interface that provides colourful and easy viewing of sensor data.
- Support for accessing a handful of the most common vehicle sensors.
- Ability to read malfunction codes from the automobile computer.

Development of NVision Technologies *Automotive Diagnostic Tool* is well under way based upon the designs discussed in this document.

Table of Contents

List of Figures

Glossary

CSA	Canadian Standards Association
ISO 9141	International Standards Organization OBD-II communication mode, used by Chrysler and most foreign cars. One of three hardware layers defined by OBD-II
J1850	An SAE Standard set in March, 1998 on Verification Test Procedures. One of three hardware layers defined by OBD-II.
J1962	An SAE Standard set in June, 1992 on Diagnostic Connector
J2178	An SAE Standard set in January, 1994 on Network Messages
J2190	An SAE Standard set in June, 1993 on Enhancing E/E Diagnostic Test Modes
OBD-II	On Board Diagnostics 2. A standard for interfacing with all automobiles manufactured after 1996.
PCM	Powertrain Control Module. The most sophisticated form of the automobile computer that electronically controls all drive functions from fuel management to gear selection for an automatic transmission.
PWM	Pulse Width Modulated
RS-232C	Recommended Standard-232C, a standard interface approved by the Electronic Industries Alliance (EIA) for connecting serial devices.
SAE	Society of Automotive Engineers
UART	Universal Asynchronous Transmitter Receiver. Hardware to send and receive serial without a common clock signal.
UI	User Interface
USB	Universal Serial Bus
VPW	Variable Pulse Width

1 Introduction

The automotive diagnostic tool will interface directly with the electronics on newer vehicles, in order to deliver a wide range of information about the internal operation of the automobile, to the user. This link is made possible by connecting to the automobile's central computer via the OBD-II standard interface. This protocol gives the hobby mechanic access to real-time data from sensors that are distributed throughout the vehicle, information about potential malfunctions detected by the computer, and the ability to reconfigure the internal computer.

Our project will be completed in a series of stages, with each one building closer to a production-quality unit. The first stage's final deliverable is a proof-of-concept device produced for ENSC 340, which is slated for completion by mid-December, 2003.

1.1 Scope

This document describes the designs that engineering staff at NVision Technologies has proposed for the automobile diagnostic tool. These designs fulfill the functional requirements, as specified in the NVision Technologies' *Functional Specification for an Automotive Diagnostic Tool*. The scope of this document covers all aspects of the project including: hardware, firmware, software, and system design. As well, a comprehensive test plan has been included.

The proof-of-concept stage is fully defined herein, and any further changes to our functional specifications will require additional changes to this document. Substantial development work will be required to bring this product to market, so it is expected that final designs will evolve with experience gained from earlier designs.

1.2 Referenced Documents

[1] Proposal for an Automotive Diagnostic Tool. NVision Technologies.

[2] Functional Specification for an Automotive Diagnostic Tool. NVision Technologies.

[3] SAE Standard J1850: Class B Data Communication Network Interface

[4] SAE Standard J2178: Class B Data Communication Network Messages

[5] SAE Standard J2190: Enhanced E/E Diagnostic Test Modes

[6] Microchip Technologies Inc. PIC18F252 Datasheet

[7] Respective manufacturers datasheets

1.3 Intended Audience

Design engineers will use this document to create design specifications for components of the automotive diagnostic tool.

Financial and marketing representatives (entrepreneurial engineers) will use this document to secure venture capital and other sources of funding for prototype development.

Quality engineers will use this document to verify the final product has met intended specifications.

2 System Requirements

2.1 System Overview

The diagram shown below shows a qualitative system overview of the product. The PCM collects data from various sensors throughout the car and performs various diagnostic and control functions. The user can request data via the graphical User Interface, which then is relayed to the PCM via the translator cable and returned along the same route. All user interaction occurs at the User Interface, while the translator cable connects to a port on the car, located within 2 feet of the steering wheel.

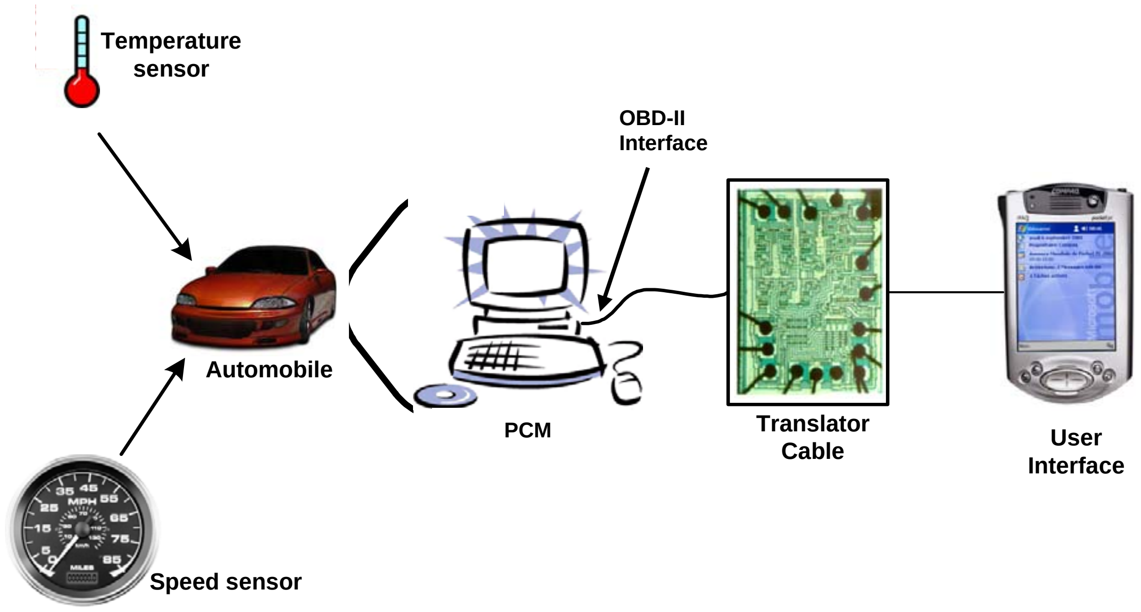

Figure 1 – System Overview

As shown in Figure 1, the entire system is composed of three modules:

1) OBD-II Interface,
2) Translator Cable, and
3) User Interface Unit

To implement this system, Figure 2 shows the detailed communication path between various components.

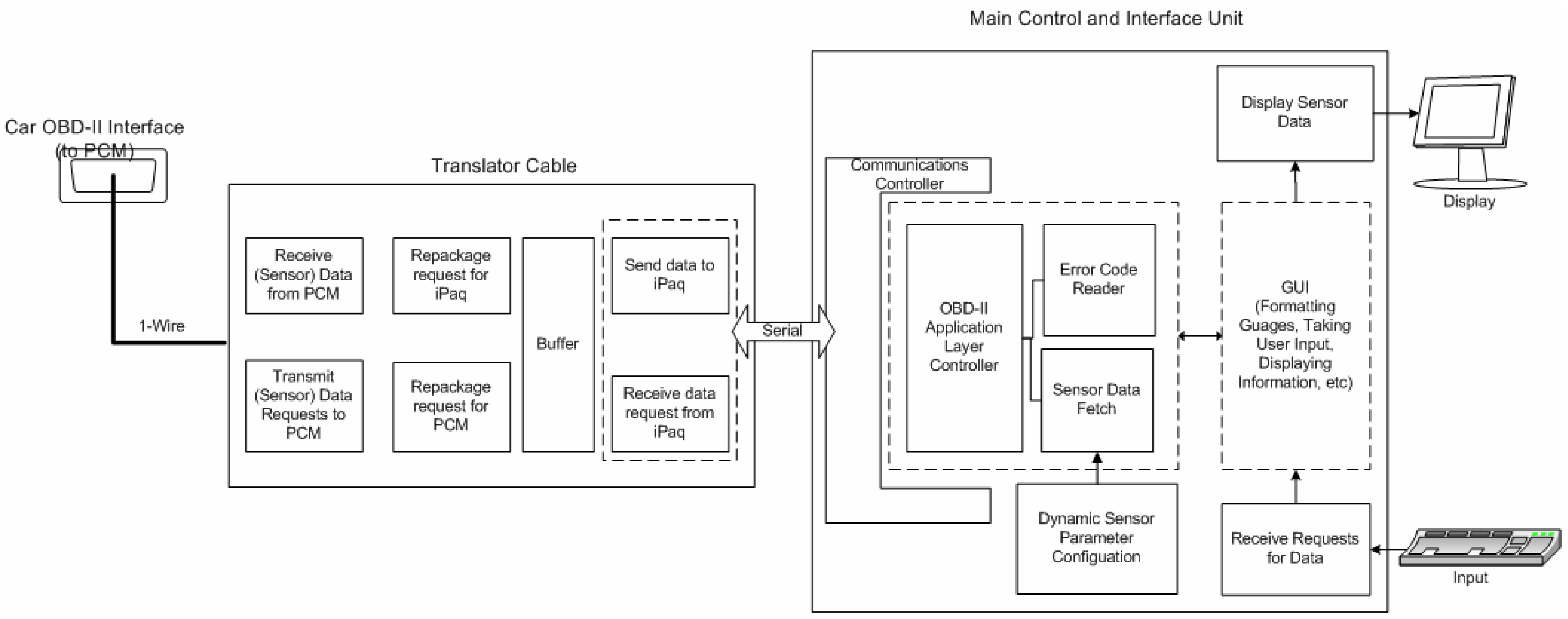
Main Control and Interface Unit
Car OBD-II Interface (to PCM)
1-Wire
Translator Cable
Receive (Sensor) Data from PCM
Repackage request for iPaq
Buffer
Send data to iPaq
Transmit (Sensor) Data Requests to PCM
Repackage request for PCM
Receive data request from iPaq
Serial
Communications Controller
OBD-II Application Layer Controller
Error Code Reader
Sensor Data Fetch
Dynamic Sensor Parameter Configuation
Display Sensor Data
GUI (Formatting Guages, Taking User Input, Displaying Information, etc)
Receive Requests for Data
Display
Input

Figure 2 – Block Diagram of System Design

When the main control unit receives a data request from the user, the communications controller sends a serial packet to the Translator Cable. The Translator Cable then repackages the request, and transmits the sensor data request to the PCM. After processing the request, the PCM sends the requested data to the Translator Cable. After proper repackaging, the Translator Cable sends the to the communications controller via the serial connection where, if no error code was detected, the sensor data will be processed by the User Interface.

2.1.1 Reliability and Autonomy

The Translator Cable component of the overall design will have removable interfaces to both the User Interface and the PCM, and will act as a stand-alone unit. The system will have plug and play capabilities that deal with a user removing from the Translator Cable the connection to either the OBD-II port or to the User Interface. The OBD-II port/Translator Cable connection will be controlled by code from the firmware side, while software from the User Interface will control the User Interface/Translator Cable link. The mechanisms built into the firmware and software will act so as to minimize the possibility of crashes and failures, by directing traffic in such a way that both interfaces are always available to handle the load.

3 System Hardware Design

This section details the hardware components chosen for the design of the automotive diagnostic tool. Special attention is paid to some of the off-the shelf components used in the proof-of-concept stage, which will be also be implemented in hardware for future production models.

The figure below shows a basic block diagram of the major hardware components.

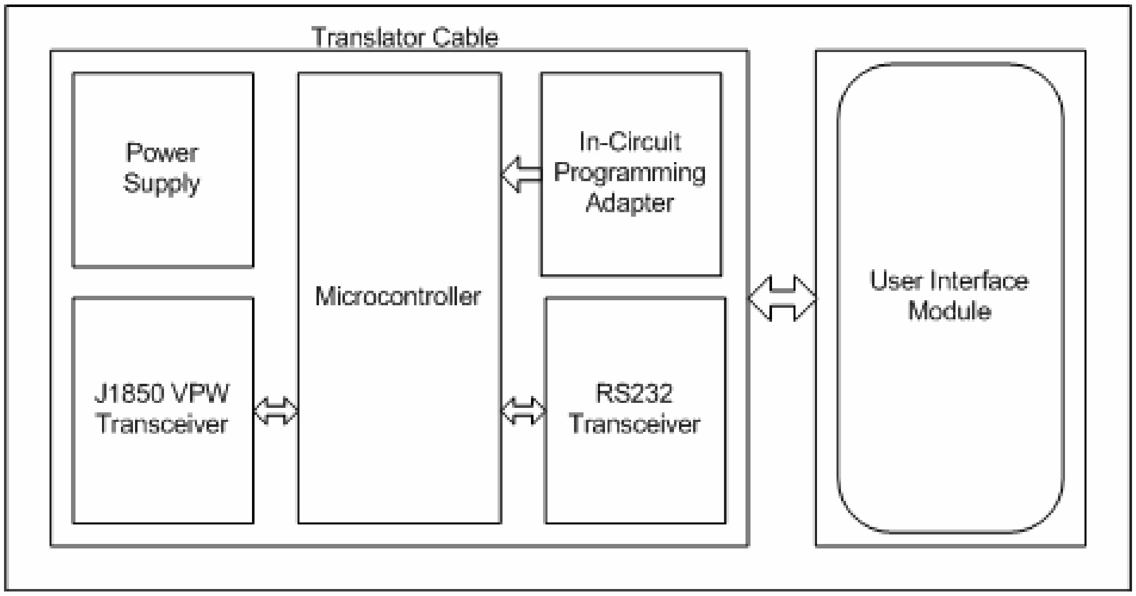

Figure 3 – Hardware Block Diagram

The User Interface Module is a stand-alone unit that provides the complete user experience, including a colourful graphical interface (GUI) as well as communication to the translator cable, and data storage as required.

The translator cable consists of several peripheral blocks surrounding the microcontroller at its centre. The microcontroller is responsible for controlling the flow of data between the User Interface and the PCM by buffering transmissions and encapsulating J2190 messages (received from the UI) into J1850 VPW packets for sending to the PCM. The microcontroller will be solely responsible for all timing, conversion, and error-checking duties. The two transceiver blocks perform level conversion from the CMOS-compatible levels at the microcontroller to levels at their respective interfaces, while the in-circuit programming adapter merely allows the microcontroller to be reprogrammed without disrupting the physical circuit.

3.1 User Interface Module

In order to save considerably on development time and financial expense, an off-the-shelf User Interface module was selected. In particular, the latest personal digital assistants (PDA) sport high-resolution colour displays, fast processors, and ample on-board memory. For this design, a Compaq iPaq 3650 was selected because it was available inexpensively, and some group members have experience with the device. The device provides more than enough horsepower with the following features:

- 240 x 320 pixel display with over 4000 colours.
- User Input via touch screen.
- 32 MB of RAM shared between program and data memory.
- Built-in RS232 serial and USB ports.
- 200 MHz processor.

3.2 Microcontroller

The microcontroller forms the core of the firmware design. To meet performance requirements and provide easy prototyping, the device had to meet or exceed the following requirements:

- 16 kB of Electrically Erasable Programmable Program memory.
- In-circuit programmable without specialized hardware.
- 1 kB on-board RAM to buffer data and run code.
- On-board UART peripheral for communications with UI.
- On-board Capture and Compare (CCP) peripheral for J1850 timing requirements.
- Sufficient speed to meet real-time requirements (~ 10 MIPS).
- Readily available C-Compiler.

The PIC18F252 from Microchip Technologies exceeds all of these requirements, and was available free of charge from the manufacturer. In addition, some group members have extensive experience with this family of devices. A schematic for the basic configuration is shown in the following figure.

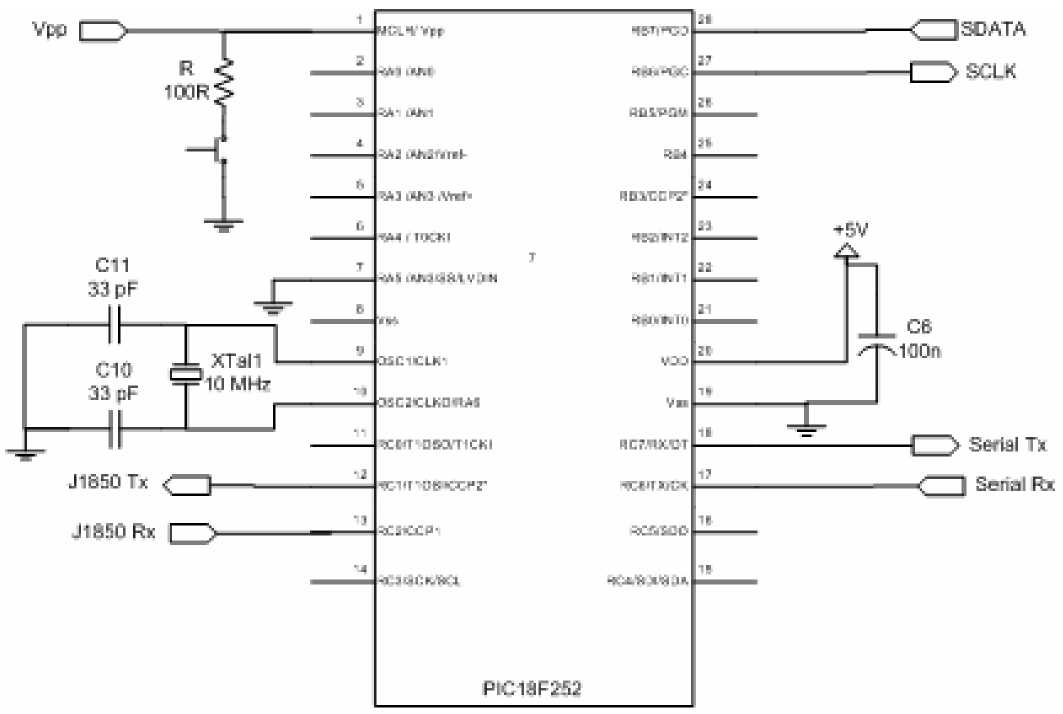

Figure 4 – Microcontroller circuit schematic.

The system clock is provided by a 10 MHz crystal, while the PIC is configured to internally multiply this value by 4, thereby providing the required throughput of 10 MIPS (PIC architecture requires 4 external clocks for each internal instruction). Additional peripheral circuitry connects directly to the microcontroller pins, as shown above.

3.3 RS232 Transceiver

The role of the RS232 transceiver is to convert between the CMOS-compatible voltage range of 0V ~ +5V and the RS232-C standard -25V ~ +25V. For the purposes of this design, the device should accomplish this conversion with a single supply rail that is also shared with logic power (i.e. +5V) for convenience. The MAX232 and related products from Dallas-Maxim Semiconductor provide this functionality with few external parts. The schematic for the circuit used in our design, as derived directly from product data sheets, is shown in the figure below.

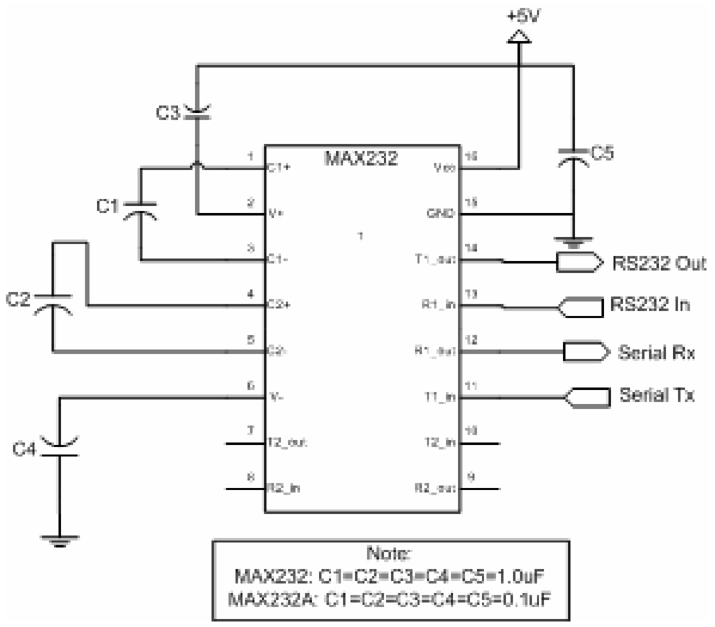

Figure 5 – Schematic diagram for RS232 transceiver circuit.

Capacitors C1-C5 provide storage for the internally-controlled charge pump, which generates the required voltages for the RS232 side of the circuit. The MAX232 contains two transmitter (CMOS to RS232) and two receiver (RS232 to CMOS) ports, of which one port from each is used to transmit and receive data. The remaining two ports could potentially be used to implement a hardware flow-control system between the Translator Cable and iPaq, if the software and firmware were modified accordingly.

3.4 J1850 VPW Transceiver

As with the RS232 interface, the J1850 VPW bus uses signal levels (0V -12V nominal) that are incompatible with microcontroller inputs and outputs, thereby necessitating some external transceiver circuitry. The J1850 bus is also a single-wire bus, so a mechanism is needed to separate the single line into the separate transmit and receive paths required for the microcontroller firmware. In addition, the J1850 specification places strict requirements on the minimum rise and fall times of signals transmitted on the bus, which means wave-shaping filters need to be included in this circuitry. Although the required circuitry would be straightforward to implement using discrete components, large semiconductor manufacturers offer highly efficient integrated solutions such as the AU5783D from Philips Semiconductor. A schematic of the prototype circuit, as derived from product documentation, is shown in the figure below.

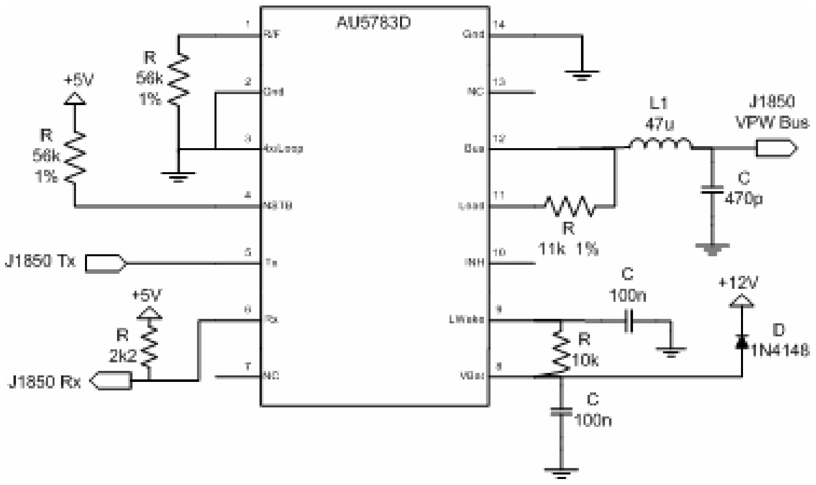

Figure 6 – Schematic diagram for J1850 transceiver circuit.

Certain features, such as power supply control and internal shutdown are not implemented for this prototype, in order to simplify the circuit. Note that the receive port open-collector output is pulled up to the logic high voltage.

It should also be noted that we also received this part free of charge, which proved critical in the selection process.

3.5 In-Circuit Programming

The programming connections are provided to allow easy configuration of the PIC18F252 on-board program memory while in the application circuit. In-circuit programming will significantly reduce firmware and software development time, and also allow the unit to be easily upgraded.

An external programmer is still required to be connected in order to provide necessary programming voltages and buffering. A schematic of this configuration is shown in the figure below.

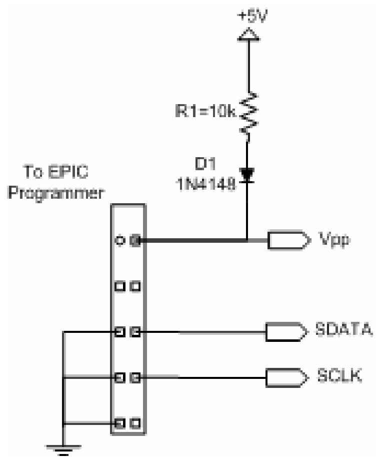

Figure 7 – In-Circuit programming header.

3.6 Power Supplies

The power requirements for this prototype are fairly relaxed, allowing a simple solution to be implemented. More specifically, the current draw from the +5V supply rail, for each module, can be summarized as per the following table.

Table 1 – Power consumption summary

Module	Nominal Current Draw (mA)
PIC18F2F2	25mA (operating) + 15mA (I/O ports) = 40 mA
RS232	30 mA
J1850	15 mA
Total	**85 mA**

With such a small current being drawn, a simple linear regulator circuit can be used to step down the 12V supply from the automobile battery, to the usable 5V rail. A standard 7805 regulator was selected and used in the configuration shown in the following figure.

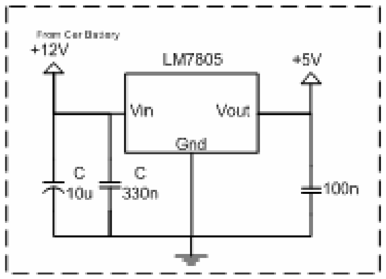

Figure 8 – Power regulator circuit.

This device is capable of supplying well over the 100 mA that our circuit requires, without significant power dissipation problems. All capacitors shown in the circuit are to filter and bypass, to improve line regulation.

4 System Firmware Design

The primary function of the firmware is to act as a middle layer for data transmission between the User Interface and the PCM. This translation is accomplished by receiving packets from the user-interface (UI). The translator cable repackages this information into packets acceptable for transmission on the automobile's J1850 single-wire bus. Our cable also controls the appropriate timings to transmit along that bus. Reception follows in a reverse manner.

The translator cable merely translates between two distinct and incompatible data-link standards without any knowledge of the data passing through.

4.1 Firmware Design Overview

The following diagram outlines the firmware design for the translator cable.

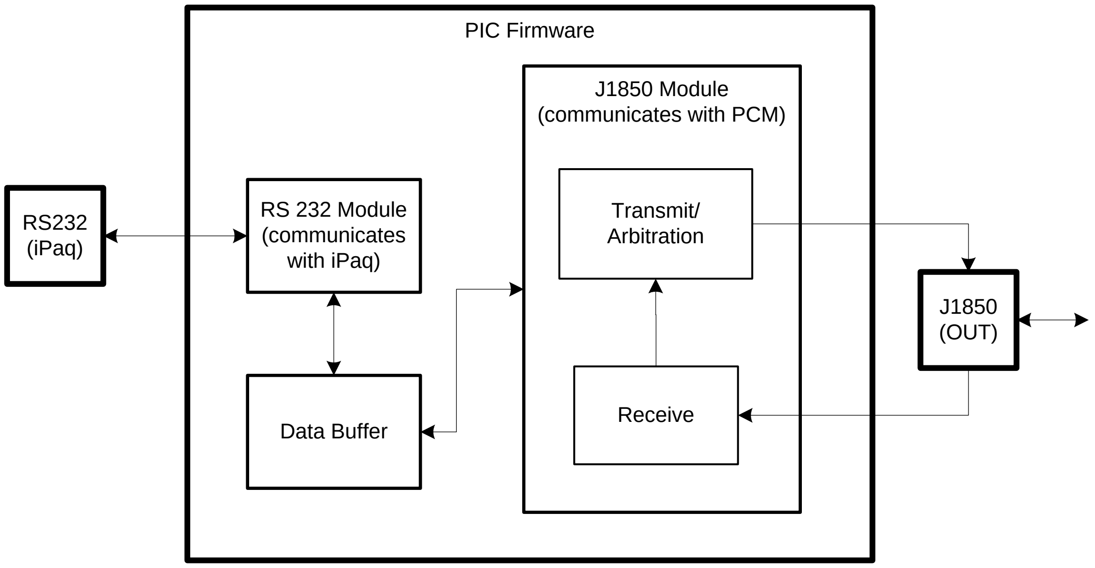
PIC Firmware
J1850 Module (communicates with PCM)
RS232 (iPaq)
RS 232 Module (communicates with iPaq)
Transmit/ Arbitration
J1850 (OUT)
Receive
Data Buffer

Figure 9 – Firmware System Overview

The firmware is broken down by its interfaces, specifically the RS232 interface to the iPaq, and the J1850 interface to the PCM. The two interfaces act independently of each other, as data is asynchronously received from the iPaq, and data is asynchronously sent to the PCM. Data from both interfaces is buffered sufficiently to prevent errors due to slow communications.

4.1.1 Interfaces with the Translator Cable

The translator cable interfaces with our User Interface and the PCM. The cable will interface with the User Interface via a common RS-232 connection. On the other side of the cable is the J1850 bus. The J1850 bus is a multi-node, master-less system, which means multiple devices must be able access the bus without a central node to control activity. Each node, including the translator cable, must self-arbitrate their access to the bus, which is handled by the *Arbitration* block within the transmitter signal path. The arbitration process is further described in Section 4.2.

4.1.2 Real-Time Considerations

Due to the nature of the device, many of the activities occurring within the firmware are triggered asynchronously. For example, the timing user input cannot be predicted, nor can responses from the PCM itself. Furthermore, implementing such a low-level communications standard makes the implementation of this system timing-critical. As such, real-time embedded system techniques are being employed to ensure all processes described above are able to run when needed and as fast as needed within a reasonable workload. Details of this real-time emulation are described in Section 4.5.

4.2 Communications with PCM

The main objective of the J1850 Module is to send iPaq requests for sensor data and receive the corresponding sensor data from the PCM. The following diagram outlines the lower level firmware that is designed to communicate with the PCM.

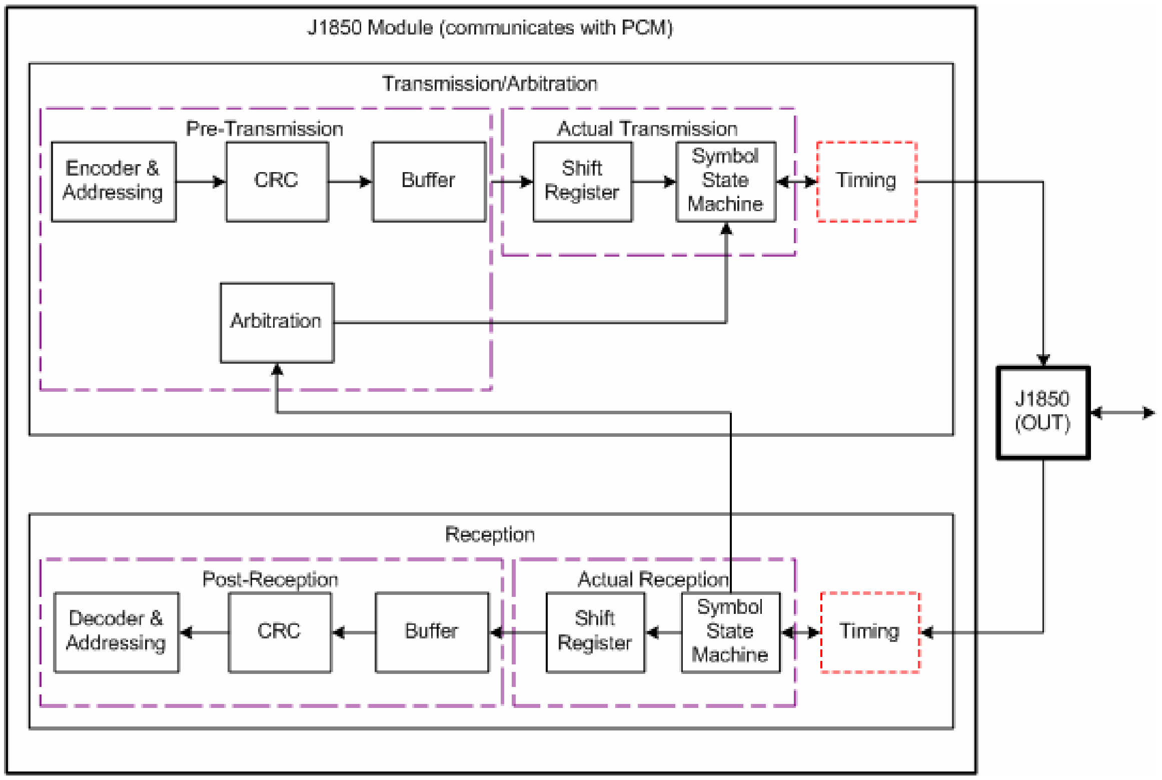

Figure 10 – J1850 Module Overview

The translator cable communicates with the PCM via the OBD-II port on the car, through the Variable Pulse Width standard. This standard is fully defined in the SAE J1850 specification, and the firmware fully follows this standard.

4.2.1 Pre-Transmission and Post-Reception

Prior to transmission to the PCM, the J1850-Module repackages the sensor data request from the iPaq by creating a new header, as per the SAE J1850 Standard. CRC checks are generated, and the new header byte is then saved in a buffer for transmission.

After having received information, the data is buffered, the CRC checks are decoded and performed, to ensure correct data reception, and the header information is decoded and repackaged to the iPaq.

4.2.1.1 CRC Checking

CRC checking is an essential part in implementing the J1850 standard because it ensures the correctness of the data received. The CRC byte can be found by the following Modulo 2 division equation:

$$\frac{X^{8*}D(X) + X^{n} + X^{n+1} + ... + X^{n+7}}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$$

D(X) is the data packet being sent, n is the length of the data packet in bits, and P(X) is the CRC division polynomial $X^8 + X^4 + X^3 + X^2 + 1$ (11D in hexadecimal). The X^{8*} means to left bit shift the data D(X) 8 times, so that the most significant bit of D(X) lines up with the most significant bit of the $X^n + \ldots + X^{n+7}$. In Modulo 2 arithmetic, each digit is considered independently from its neighbours, which means numbers are not carried or borrowed. A series of exclusive Ors (XOR) can be used to perform this calculation. When transmitting, the CRC byte is made to equal to $\overline{R(X)}$, where $\overline{R(X)}$ is the 1's complement of R(X). On receipt of new data, the CRC for the entire packat (including transmitted CRC) is calculated. Then, if $\overline{R(X)}$ is equal to a unique constant (C4 in hexadecimal), no error has been detected and the data packet is considered correct.. Otherwise, the data segment contains invalid information and an error flag must be set to indicate this error. The firmware will then know that transmission was unsuccessful and act accordingly.

4.2.1.2 Arbitration

In order to transmit over the single-wire bus, arbitration is required. The arbitration process begins by listening to the J1850 bus. If the bus is idle, and if the translator cable needs to transmit information, then it will transmit the first bit on the line. Bit-by-bit, the line is checked to verify that the cable has correctly sent the last bit, and if so, it will continue sending the information. At any point in this bit-by-bit arbitration, if the bit on the line is not the last bit sent by the translator cable, then the firmware understands that the PCM, which has a higher bus priority, is utilizing the single-wire bus. The Translator Cable will then abort the transmission and service the signal sent by the PCM. The firmware will try to send the entire packet again when the line becomes idle.

4.2.2 Transmission and Reception

The data transmission and data reception is achieved serially over a single wire using variable pulse-width (VPW) modulation as specified by the J1850 standard. Detailed timing issues need be established to allow the Translator Cable to communicate effectively with the PCM.

Along the receive path, the *Timing* block uses the PIC's Input Capture peripheral to measure the period between transitions on the bus (Note: J1850 VPW is set up such that every bit transmitted requires only a single bus transition, from low to high or vice-versa). The *Symbol State Machine* then uses this symbol (the combination of bus logic level and the duration of the pulse), along with the previous bit value, to decode the value of the received bit. As each bit comes in, it is shifted into a storage register, which then forms the bytes of incoming data.

Transmission is the reverse of reception. The *Shift Register* shifts out data bits one by one to the *Symbol State Machine*, which then decides the required duration of the next pulse. This duration is then used to configure the *Timing* block, which uses the Output Compare peripheral to ensure precise timing for each pulse. The J1850 VPW bus value toggles to an opposite value for every bit, so the *Timing* block is configured to do the same, and merely toggles the output to a different value when the pulse has reached its required duration.

4.3 Communications with User Interface

The RS-232 module is designed to interface with the iPaq User Interface system by receiving requests and sending sensor data, in the form of NVision Technologies' header and over a standard serial connection. The following diagram outlines the firmware designed to communicate with the iPaq.

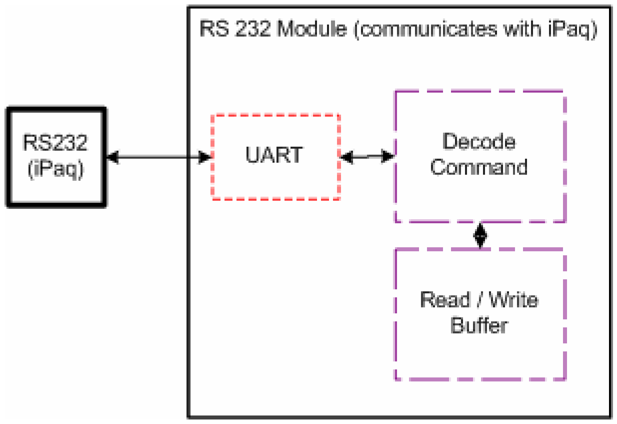

Figure 11 – RS-232 Module Overview

The operation of this module can be broken down into the physical UART, the command decoder, and the data buffer. Note that, initially, no error checking will be included for this interface so that code can be further simplified.

4.3.1 Universal Asynchronous Transmitter Receiver

The selected PIC18F252 supplies a hardware UART to handle the shifting of data across the serial lines. The peripheral will be configured to generate interrupts on the receipt of data, from which point the incoming data is promptly stored in a safe buffer to allow further data to be received. This method fits well with the asynchronous design model used throughout the rest of the firmware.

For this initial design, hardware flow control will not be used to control transmission along the RS-232 lines.

4.3.2 Command Decoder

As described later in the software design section, the translator cable supports several modes of operation. The UI will inform the cable that it should enter these modes via commands sent across the serial (RS-232) link. It is the task of the command decoder to interpret incoming data and respond accordingly. Whenever a byte is received, the decoder will determine whether it is a command or data, and deals with it appropriately. The following flow diagram illustrates the process.

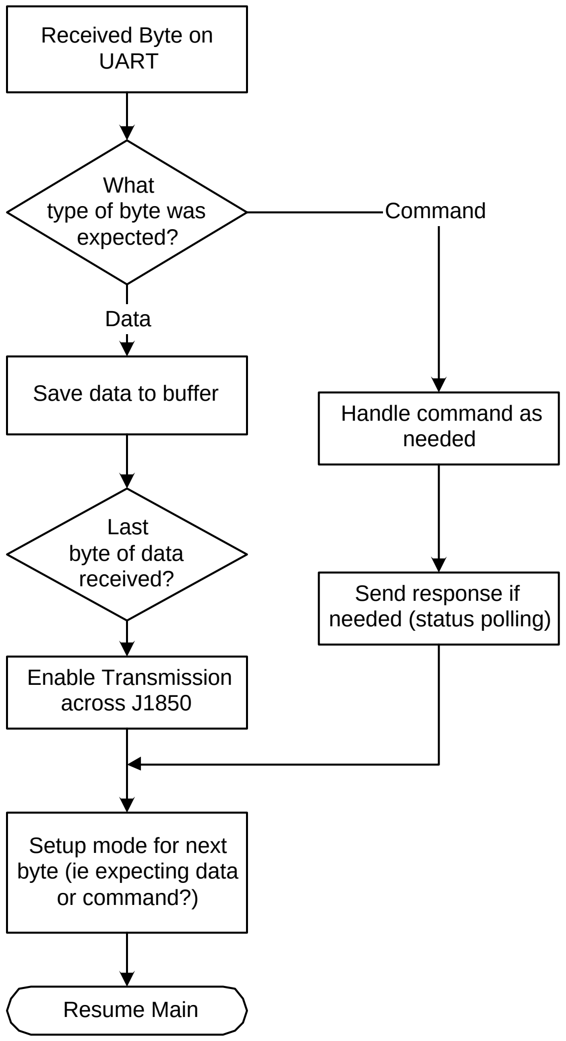
Received Byte on UART
What type of byte was expected?
Command
Data
Save data to buffer
Handle command as needed
Last byte of data received?
Send response if needed (status polling)
Enable Transmission across J1850
Setup mode for next byte (ie expecting data or command?)
Resume Main

Figure 12 – Program flow for UART command decoder.

4.4 Data Buffers

A large role of the translator cable is to alleviate the UI of timing-critical activities, and thereby acts much like a buffer. To accomplish this high-level buffering, a robust internal buffering mechanism is required. Internally, a fixed block of memory is permanently allocated to separate transmit and receive (i.e. to and from the PCM, respectively) buffers, where the receive buffer should be fairly large (more than 1 kB) to accommodate the possibility of the PCM returning frequent data that the User Interface cannot handle. The transmit buffer need not be so large because the UI is capable of throttling down its requests. When data arrives at the cable from either interface, the respective modules request storage space from the buffer, where it remains until the opposite interface is able to send it away.

4.5 Real-Time Considerations

In order to meet the real-time requirements of our system, the firmware must be structured such that data is processed before it is required so that information is always ready when asynchronous events are triggered. For example, an internal timer that triggers an interrupt will handle bit timing for the J1850. When this interrupt handler is called, the value for the next timer interval must already be calculated to allow a seamless transition. All of this processing and calculating will be done in a constantly looping main program, while outside events trigger interrupts that make use of the calculated data. Figure 13 outlines the main program (sequential) flow.

This flow may be interrupted at any time by either of the interface ports to request data, so care must be taken ensure processing speed is fast enough. The sleep cycle is tentatively schedule if processor time permits, in order to save power.

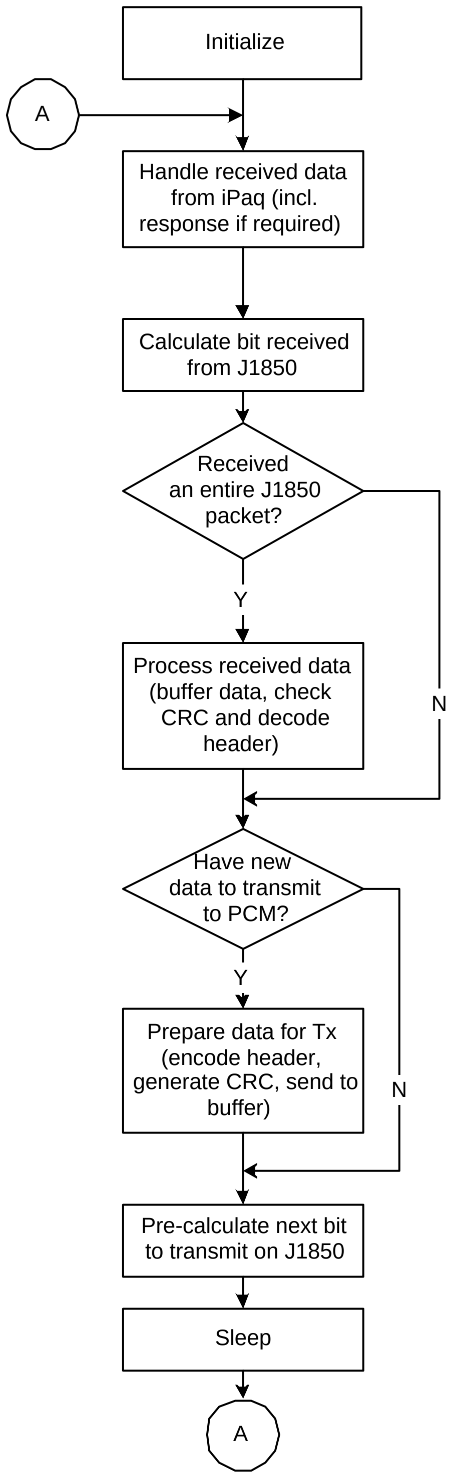
Initialize
A
Handle received data from iPaq (incl. response if required)
Calculate bit received from J1850
Received an entire J1850 packet?
Y
N
Process received data (buffer data, check CRC and decode header)
Have new data to transmit to PCM?
Y
N
Prepare data for Tx (encode header, generate CRC, send to buffer)
Pre-calculate next bit to transmit on J1850
Sleep
A

Figure 13 – Main Program (Sequential) Flowchart

5 Application Software Design

The application software will be used to interface with the PCM. As such, it must control all user interactions as well as perform the necessary encoding needed to send J2190 commands to the translator cable, and through the cable to the PCM.

5.1 Platform

NVision Technologies has chosen to write the application software using Embedded Visual C++ for its ease in GUI implementation and event driven architecture. The entire embedded tool suite is available free of charge from Microsoft and includes the software development kit (SDK) for the operating system used on this iPaq (Windows CE 3.0 PocketPC 2002).

5.2 System Features

The application software will be a robust system that will allow the user to interact with the PCM through the translator cable. The program will transmit and receive data to the translator cable via the serial port and will have the appropriate encoding and decoding to follow the J2190 and J2178 standards.

Application specific features to be included in the software:

- System query to translator cable will request status on buffers, stack and check if it is connected to the PCM. to be implemented using a Header Byte. This query will be implemented via a single byte at the front of the command sent to the iPaq, and will also signal if the attached data is a PCM command.
- Dynamic command table look-up will allow the user to specify commands to the car if the user knows the appropriate command codes and parameter information. The command table will also allow the user to specify how the diagnostic tool should interpret the incoming data.
- Animated graphs to display engine temperature, engine efficiency and other sensor data.

5.3 High Level Implementation

The high level implementation of the software will occur in three distinct stages:

- The Graphical User Interface
- Data Transmission
- Data Acquisition

Each of these stages presents unique challenges to the software designers and deserves special attention in the design documentation.

5.3.1 Graphical User Interface

The graphical User Interface will be the only PCM access point for the local user. Its purpose is to create an environment that easily allows the user to request and interpret data from the diagnostic tool. In the proof-of-concept stage, the focus of the graphical User Interface will be to manage the flow of data between the translator cable and the user in an easy to use fashion. While effort will be taken to create a user-friendly design, functionality will take precedence over visual flavour.

The GUI for the automotive diagnostic tool will consist of separate window layouts for each of the different major features. The windows will share a similar feel, but be tailored to serve each feature's individualistic purpose. Wherever appropriate, graphics will be used to display data in an easily to discern fashion. The focus of the graphical User Interface will be on creating a program that allows the user to traverse the small screen windows intuitively.

5.3.1.1 Main Software Functions

NVision Technologies foresees three main software interactions that the automotive diagnostic tool will adopt: sensor acquisition, diagnostic trouble-shooting and PCM parameter modifications.

Acquisition of specific sensor data will be the main feature of the proof-of-concept stage software. Specific sensors will be polled and the requested data will be presented to the user in a user-friendly format. Sensor acquisition on the automotive diagnostic tool will support different data formats including Boolean and continuous values.

The diagnostic trouble-shooting and PCM parameter modification abilities are functions that are not critical to the application software but are rather add-on features. The trouble-shooting mode would allow the user to gain more insight on different error warnings that a car has such as the ambiguous *check engine* and *maintenance required* lights, while the ability to modify PCM parameter information would be a great feature for hobby enthusiasts who want to get maximum performance out of their vehicles.

5.3.1.2 Graphical Features

Whenever possible, the use of tables and text will be kept to a minimum, replaced with graphics and charts to display sensor data. The diagnostic application will be written as a small screen application for a PDA and must therefore manage the flow of information in the most efficient manner through the use of graphics and text.

In particular, gauges, which appear similar to those on a typical dashboard, will be used to indicate sensor values such as speed, oil pressure, and intake air temperature.

Furthermore, a graph will be plotted to display the calculated horsepower vs. speed (revolutions per minute) curve. These display will be created as a custom controls, derived from classes available from the Microsoft embedded toolkit.

5.3.1.3 Dynamic Command Structure

Communications with the translator cable will occur over a serial connection utilizing the SAE J2190 standard. To create the appropriate commands to diagnose every sensor and function available inside the PCM would waste a considerable amount of memory. Instead, the automotive diagnostic tool will support a file-parsing process that will encode generic commands in the J2190 standard. The process will allow the user to specify the 2190 command type, parameter information, and how to handle the returning data in a command file which can be easily appended to. This will allow the diagnostic tool to be robust and handle many different command, parameter and output display combinations. The basic structure is shown below in Figure 14.

Sensor Name	Sensor Address	Data Parameters	Data Type	Display Method

Figure 14 – Dynamic Command Structure Example

An additional benefit of such a structure is that end-users can add support for more sensors (and thus different cars) without having to directly modify source-code, thereby alleviating responsibility from NVision Technologies for maintaining the product. Prototyping and debugging will also be further simplified.

5.3.2 Data Transmission

The User Interface will interact with the translator cable via a serial connection utilizing the RS-232 standard. Commands to the translator cable will be multiple bytes long with the embedded commands meant for the PCM encoded in the J2190 standard.

The data will be transmitted asynchronously via a flow plan that NVision Technologies has devised. Part of this system is an added header byte to the J2190 commands that the User Interface sends out. This header byte allows the User Interface to communicate with the translator cable such as requesting system status, identifying whether or not the translator cable is connected to the OBD-II port, signalling the translator cable that the attached command is to be passed to the PCM, and to diagnose any translator cable errors.

Once the User Interface has created an appropriate command code, the command is sent to the translator cable all at once. The User Interface does not wait for a response from the translator cable but rather assumes that the command was correctly sent to the translator cable's incoming data buffer. Once a command has been sent over the serial connection, the User Interface can immediately send a new command.

5.3.2.1 NVision Header Byte

NVision Technologies foresees the need to create a header byte to all commands sent to the translator cable via the serial RS-232 connection. If a PCM command is to be sent, the J2190 command structure that the User Interface sends to the translator cable will be appended to the header byte. This additional byte will be decoded and serviced by the translator cable. It will signal to the translator cable the nature of the command such as whether or not the command is meant for the PCM or if it is meant for the translator cable to service by itself. The additional byte will not be passed to the PCM; it will be stripped if the remainder of the command will be sent to the PCM via the OBD-II interface.

5.3.2.2 J2190 Encoding

The Society of Automotive Engineers (SAE) specifies the application layer commands that the User Interface follows to transmit data and requests to the PCM via the translator cable. The J2190 standard [5] and the J2178 standard [4] are the documents that NVision technologies is following in implementing this layer.

5.3.3 Data Acquisition

When communicating with the PCM via the Translator Cable, the User Interface specifies the command using the J2190 standard and sends it out over the serial connection. If the data transmission occurred correctly, the User Interface expects a response from the PCM.

The design of the User Interface and the Translator Cable is such that the User Interface is never locked into waiting for a response from the PCM. When a response is sent from the PCM and processed by the translator cable, it is first stored in the translator cable's output buffer and a signal is sent to the User Interface signalling that the translator cable is ready to transmit data.

When able, the User Interface signals the translator to begin sending the data over the serial connection. The data is stored in an input buffer waiting for the entire command to arrive. Responses from the PCM do not necessary arrive all at once. The data sits inside a data buffer until the User Interface can verify that the entire response has arrived and it can then be decoded and serviced by the User Interface.

5.3.3.1 J2190 Decoding

When a command has fully been received from the translator cable, the User Interface can then decode the signal and interpret the data. The User Interface will follow the appropriate error checking functions and communicate any problems with the translator cable by re-requesting data that may have been incorrectly transmitted.

6 Test Plan

Each of the three different components will be tested individually. Once each component has been verified, system-wide testing will take place to make sure that interaction between the various sub-systems is co-ordinated and efficient. These test plans detail some of the testing that will be completed. As the design process proceeds, additional tests will be appended to this list.

6.1 Hardware Test Plan

The hardware testing begins as soon as the parts arrive. Each of the parts will need to be tested to ensure that they operate as specified. The following tests will be used to ensure the quality of the hardware.

6.1.1 Power Supplies

Because three different voltage levels will be used to power the different components in the automotive diagnostic tool, it is important to maintain consistent and uniform power. The power that will feed the microprocessor, the RS-232 transceiver and the J1850 module will be fully tested to make sure that the power being supplied remains within the specified tolerances under full load conditions (transmitting at full data rate on both interfaces) and under idle conditions (the microcontroller is in sleep mode).

6.1.2 Translator Cable Interfaces

Every connection to the translator cable will be tested to ensure the following:

1. The in-circuit programming connection is working properly, and the PIC is able to accept programming. Loading code and verifying its operation, as well as reading the code from the chip, will accomplish this test.
2. The J1850 VPW Transceiver is functioning correctly, and the Translator Cable is able to send and receive signals to the J1850 VPW Transceiver port. This will be tested by configuring a constant pulse-width output from the microcontroller, and measuring voltage levels, and rise and fall times on the J1850 connection, via an oscilloscope.
3. The RS-232 module is functioning correctly, and the Translator Cable is able to send and receive serial signals through the RS-232 UART. This test will be performed by echoing bytes received from a test computer and verifying their values.

6.2 Firmware Test Plan

The testing of the firmware will be on a continual basis, as soon as the hardware has been implemented and tested. Tests will need to be run to ensure that the firmware is able to handle the various amounts of information at both the PCM and iPaq interfaces.

6.2.1 J1850 and NVision Header Byte Encoding/Decoding

Tests will need to be done to make sure that the translator cable correctly decodes and encodes the headers to and from the PCM via the J1850 bus. Similar tests will be performed on the encoding and decoding of the NVision header byte. Tests will be also run to ensure that the CRC checking is performed correctly, and that the information sent from the iPaq is the same information requested from the PCM.

6.2.2 RS-232 and J1850 VPW Timing Testing

Part of the firmware testing will be to ensure that the transmission of signals between the PCM and Translator Cable and the User Interface and Translator Cable occur correctly. Special attention will need to be paid to the timing issues of the VPW transmissions as well as the transfer rates of the serial connectors.

6.2.3 Stack Errors and Buffer Overflow Issues

Because the firmware needs to perform in real-time, there is the chance that stack errors and buffer overflows may occur. Testing will need to be done to ensure that the system correctly handles the data associated with these problems.

6.3 Software Test plan

Testing of the User Interface will occur on several levels. Tests will need to be run to confirm that User Interface and all its sub-systems are running as expected.

6.3.1 Window Layout, Buttons and Menus

Test will be run by several user to confirm that traversing through the windows is efficient and error free. As well, users will check all the buttons and menus to make sure there are no broken links and all actions are correct and expected.

6.3.2 J2190 Standard Encoding and NVision Header Byte Testing

Tests will be done in software to confirm that the classes that are used to create the J2190 command codes and NVision Header Byte run as expected. Various benchmark commands will be compiled by hand and verified with the commands constructed in software.

6.3.3 Serial Communication Testing

Testing of the serial communication links will be done by sending out simple commands of various lengths to the translator cable in order to verify that the commands are being sent properly. Faulty transmissions will be deliberately sent to investigate whether or not the User Interface is able to recover from a serial communications error.

The tests to check the reception of the serial communications will include sending simple commands of various lengths from the translator cable in order to verify that the serial connection on the User Interface is operating correctly. Faulty transmissions will also be sent from the translator cable to the User Interface to ensure that the User Interface is correctly able to handle errors.

6.3.4 J2190 Standard Decoding and Data Handling Testing

Testing will be done to ensure that the User Interface is able to handle the different types of data being received from the PCM. Not only will the User Interface need to be able to decode the signals to gain access to the car's data, but depending on the type of data, the User Interface must also know how to respond and interpret it.

Some types of data will be stored in memory; other types will be used to create charts and graphs. The testing of the data decoding and handling will reflect this.

6.4 System-Wide Test Plan

System-wide testing will first be run solely with a PCM, and then with a PCM within a car.

A PCM compatible with the J1850 standard was purchased for proof-of -concept testing, to ensure that connecting to a live car would not be immediately necessary. NVision Technologies will connect the appropriate connectors to the external PCM and use a DC power supply to generate the required 12 Volts for this discrete device. The automotive diagnostic tool will then proceed through all levels of system wide testing.

After having ensured that the automotive diagnostic tool runs connected to the PCM, the tool will then be connected to an appropriate car. At this time, the tool will be feeding off of the 12 Volts coming from the car's OBD-II port.

6.4.1 Physical Requirements Test

Several tests must be performed to verify that our system has met the desired physical requirements. While the hand held device is being accessed everywhere inside the car, the power and the performance of the device is being monitored to make sure that the Translator Cable is long enough and the plug is secured enough to withstand some amount of external force. The device will also undergo driving conditions to be tested for its ability to withstand shock and vibration. The automotive diagnostic tool's performance during driving conditions will be compared to its performance in its stationary state.

6.4.2 System Communications Test

The overall system communications will be tested by two main stages. First stage involves hooking up the device with a test PCM that is not physically attached to the car. The test PCM has dummy sensor values that will be referred to for comparison and verification. The second stage will be hooking up the device to the actual car and perform data acquisition from the car's sensors during driving.

The first stage test can be broken into the following steps:

1. Connect the PCM, the Translator Cable and the User Interface correctly.
2. Power up the system properly.
3. Verify that the User Interface has detect a connection to the Translator Cable.
4. In the User Interface, choose "sensor display" mode.
5. Request some specific sensor data from the PCM.
6. Verify that requested information is displayed.
7. Compare the displayed information to the actual information of the PCM.
8. Repeat steps 5) and 6) as many times as needed to request various sensor data

In the second stage, all the testing procedures used in stage 1 will be repeated with the following additional steps:

9. Request data from the speed sensor and compare results to the speedometer during driving and stationary conditions.
10. Request data from the speed sensor at a different driving speed. Use the information to calculate the speed/horsepower curve and compare it to the expected resuts.

7 Conclusion

This document has outlined the design specification for the proof-of-concept *automotive diagnostic tool*. Based on these specifications, we will continue to develop our technology to fulfill a mid-December 2003 completion. This design specification provides us with a clear path to implementing our proposed system.

This design specification is well planned and researched. We have defined specific goals and have a plan to achieve them. NVision Technologies has the goals, skills set, and funding to meet these commitments.