TMC Recommended Practice

Proposed RP 240 (T) **VMRS 018-003-000**

STEEL WHEEL AND RIM REFINISHING GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to provide guidelines and inspection criteria for steel wheel and rim refinishing and for evaluating refinishing vendors. This RP describes various types of steel wheel and rim refinishing systems used today for commercial vehicle applications.

COATING SYSTEMS OVERVIEW

The following represents an overview of the three primary coating system components: primer, powder coat, and paint.

- **Primer**—The primer is an optional base coat applied to bare metal to improve adhesion of subsequent coat(s).
- **Powder Coat**—This powder coat system involves the application of an electrostatically charged powder paint that is cured in an oven. Once the wheel or rim emerges from the oven, it can be returned to service in a relatively short period of time.
- **Solvent or Water-Based Paints**—This system involves the application of a solvent or water-based paint. After applying the paint, the wheels or rims dry at room temperatures or are cured in an oven.

PROCESS GUIDELINES

1. Initial Inspection

Prior to the refinishing process, the wheel or rim should be thoroughly inspected for any damage or out-of-service conditions. Technicians should refer to TMC RP 222B, *User's Guide to Wheels and Rims*, for a comprehensive listing of conditions. Any rejected wheels or rims should be marked and tagged unserviceable and returned to the fleet if applicable.

2. Pre-Wash

Any existing dirt, grease or debris should be removed prior to the paint removal process. Failure to perform this step may have a negative effect on the coating performance.

3. Coating Removal

The primary objective in this step is to remove all of the previous coating without damaging the wheel or rim surfaces. If the date code or part number stampings are not easily legible or the surface is severely pitted after the coating is removed, then the blasting method/media may be too aggressive or the rim/wheel could be worn out. If the DOT stamp or manufacturers identifying marks are not legible for any reason, remove the rim/wheel from service and scrap.

Most processes use a blasting cabinet that uses a variety of media ranging from metal shot, to glass beads, to other material. Sandblasting or grit by itself is not recommended as it tends to reduce wheel or rim durability. Smaller, less aggressive steel shot size is recommended, typically less than S330. Blasting cabinet manufacturers recommend the most effective shot size to be used with their equipment. Machines designed to remove only solvent or water-based coatings may not effectively remove powder coat finishes without excessive surface damage. Whatever the media, it is absolutely imperative that the paint removal system results in bare metal on the mounting surfaces and bead seats. Complete removal of the old coating is recommended to facilitate inspection and refinishing.

4. Inspection Following Coating Removal

After the coating has been removed, the rim/wheel should be inspected for any cracks or out of service conditions (see RP 222B). If the DOT stamp or manufacturers identifying marks are not legible, remove the wheel from service.

5. Cleaning/Pretreatment

Following the coating removal process, some steel wheel or rim refinishing systems use solvent/alkaline cleaning solutions or other methods to prepare the bare metal surfaces for coating. Consult the coating manufacturer for recommended cleaning and pretreatment guidelines. This step in the process has a positive effect on coating performance. In order to maintain consistently clean surfaces, TMC recommends that the cleaning agent be periodically replaced.

6. Bare Wheel Handling/Storage

Technicians should only handle bare wheels with clean, lint-free, dry gloves. Bare wheels should not come in contact with the floor or anything that may contaminate the cleaned wheel/rim surfaces. Since bare wheels have no protection from moisture and other contaminants, the surface should be coated promptly to prevent flash rusting.

7. Primer Application

Aftermarket processes that include a primer or base coat may improve the performance and durability of the finish. Check with the coating manufacturer for recommendations on the use of primers or base coats.

8. Finish Application

Regardless of the type of finish being applied to the wheel or rim, it should be no more than 3.5 mils total thickness on the mounting surfaces. Coating thickness should be measured mid-way between the bolt holes in a minimum of five locations on both sides of the wheel. It is extremely important to avoid any runs or excessive coating thickness, especially around the bolt circle, wheel/rim mounting, or bead seating surfaces.

Excessive coating thickness can lead to loose fasteners, premature wear or wheel loss as the result of the joint settling in.

9. Curing

All powder coat steel wheel refinishing systems use an oven or other heat source to cure the coating. For air-dry coatings, the typical time required to ensure complete curing of the material is at least three days. Baking the painted wheel/rim will speed up curing time. Consult the coating manufacturer for curing specifications. Undercured coating will have the same effect as excessive coating thickness since the soft coating will be squeezed from the mounting surfaces and from under wheel fasteners when they are tightened.

10. Post Finish Inspection

After the finish has been cured according to manufacturers recommendations, the rim/wheel should be inspected for any runs or excessive coating thickness. If the DOT stamp or manufacturers identifying marks are not legible, remove the wheel from service.

PROPSPECTIVE VENDOR CHECKLIST

The following checklist of questions may be used to help evaluate refinishing vendors:

- Is each wheel or rim thoroughly inspected in a well-lit area by a trained technician prior to refinishing?
- Does the inspector have access to TMC RP 222B as a reference for potential out-of-service conditions?
- Does the coating removal process use the correct size media and is it replaced on a regular basis?
- Does the coating removal process result in completely bare metal mounting and bead seating surfaces?
- Are the wheels cleaned prior to coating application?
- Are bare wheels promptly processed to prevent flash rusting?
- Are there quality controls in place to ensure that worn or damaged wheels are identified prior to the finish application process and removed from service?
- Does the coating application process ensure that the total thickness does not exceed 3.5 mils on the wheel/rim mounting surfaces?
- Are there quality controls in place to ensure that the proper thickness of the coating is applied to the rim or wheel?
- Can the original rim/wheel manufacturers stamps and identifying marks be read after refinishing?

Proposed RP 311A (T)

VMRS Various

COLD WEATHER OPERATION

TABLE OF CONTENTS

INTRODUCTION .... RP 311A—1
COLD WEATHER OPERATION MAINTENANCE CHECKLIST .... RP 311A—2
ENGINE COOLANT .... RP 311A—3
ENGINE, TRANSMISSION, AND DIFFERENT LUBRICANTS .... RP 311A—4
AIR STARTERS .... RP 311A—4
HOSES AND HOSE CLAMPS .... RP 311A—5
BATTERIES .... RP 311A—5
BATTERY WARMERS .... RP 311A—5
DIESEL FUEL .... RP 311A—5
FUEL WARMERS .... RP 311A—6
FUEL HEATERS .... RP 311A—6
FUEL/WATER SEPARATORS .... RP 311A—8
OIL PAN HEATERS .... RP 311A—8
COOLANT HEATERS .... RP 311A—9
ENGINE STARTING FLUID (ETHER) SYSTEMS .... RP 311A—9
AIR BRAKES .... RP 311A—11
ELECTRIC AIR INTAKE HEATERS .... RP 311A—11
FUEL-FIRED INTAKE MANIFOLD HEATERS .... RP 311A—11
AUXILIARY FUEL-FIRED CAB AND ENGINE HEATERS .... RP 311A—12
SHUTTERS AND WINTERFRONTS .... RP 311A—12
COLD WEATHER OPERATING AIDS CHART .... RP 311A—13

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

This Recommended Practice (RP) covers the proper methods for preparing a heavy-duty, Class 7-8 commercial vehicle for cold weather operation in the ambient temperature range of -40°F to 40°F (-40°C to 4.5°C). Topics covered include readying the following for cold weather operation: engine coolant; engine, transmission, and differential lubricants; hoses and clamps; air starters; batteries and cables; diesel fuel; coolant, oil, and intake manifold heaters; fuel warmers; winterfronts; and radiator shutters.

INTRODUCTION

This RP deals with problems caused by cold weather in three cold climate zones: +40 to 0°F, 0 to -20°F, and -20 to -40°F (+4.5 to -18°C, -18 to -29°C, and -29 to -40°C). This RP consolidates information on cold weather operations from a variety of industry sources. Fleet managers should recognize, however, that the most important factor affecting cold weather operation is a fleet's own maintenance program and experience regarding their operation.

The most important part of the document—the Cold Weather Operations Maintenance Checklist—appears in the first section. Following this checklist, the RP offers recommendations in 15 different areas that important to cold weather operation. The RP concludes with a Cold Weather Operating Aids Chart. References to other TMC or SAE publications are provided when applicable. TMC publications can be obtained by calling (703) 838-1763 or visit http://wwwtruckline.com/store. SAE publications can be obtained by calling SAE Member Services at (414) 776-4841.

Issued 2/1991
Revised x/xxxx

NOTE: Engine manufacturers also publish operation and maintenance manuals for cold weather operation. Following is a listing of their documents:

- Caterpillar SEBU5898-03.
- Cummins 3387266-R and 3379009-03.
- Freightliner/Detroit Diesel Engineering Bulletin 40 (18SA208, Rev. 1), Engineering Bulletin 38 (18SA0205).
- Volvo/Mack TS57284.
- International Truck and Engine Corporation, see http://www.navistar.com.

COLD WEATHER OPERATION MAINTENANCE CHECKLIST

Coolant System

1. Check antifreeze strength for adequate protection at the lowest anticipated ambient temperature. If required, add premixed coolant only.
2. Check supplemental coolant additives or water filter additive strength; do not over protect.
3. Pressure test cooling system at 15 to 18 psi after turning on heater control valves. Repair any leaks. Better results will be obtained testing a cold engine.
4. Pressure test radiator cap to the recommended pressure of the vehicle.
5. Carefully check all heater and water hoses for wear, cracks, hardening, or softening. Tighten any loose hose clamps.

Cab Heating System

1. Clean out heater screens or filters and check for any heater core obstructions.
2. Check heater fan operation using an amp meter to check current draw. Draw should be no more than the rated amps per motor.
3. Check bunk heater operation and switch condition.
4. Check heater controls for proper function. Operate in defrost and floor heat modes.
5. Check cab floor grommets for any holes and sealing capability (especially around clutch throttle, wiring, hoses, etc.).
6. Check transmission tower seal with the cab down. Replace if it is not sealing out air.
7. Check weather stripping around doors and for doors not adjusted properly. Repair as needed.

Air System

1. Drain air tanks and notify shop foreman if oil is present in drained water.
2. Remove and clean air dryer purge line and air governor signal line.
3. Check for proper vent on exhaust port if air governor equipped.
4. Check air dryer heater operation; a four to five amp draw with element cold is normal.
5. Ensure it turns off and on properly.

Batteries and Electrical System

1. Disconnect battery cables and load test each battery individually at 1/2 CCA rating for 15 seconds. (Example: 625 CCA battery rating, use 300 amps.) After one minute, measure the voltage. 9.6 volts or higher at 70°F (21°C) or 8.5 volts or higher at 0°F (-18°C) ambient temperature Indicates a good battery.
2. Check connections at starter using a voltage drop tester.
3. Check alternator voltage and current output. It should be 13.2 -14.2 volts, for a 12-volt system, when testing with a load at 120 percent of the alternator rating.
4. Check condition and tension of alternator belts.

Starting Aids

1. Check condition of block heater with an ohmmeter. A 9-15 ohm element resistance is normal for 1000- to 1500-watt, 120-volt heaters. For 240-volt heaters, the resistance is normally 36-60 ohms for 1000 to 1500 watts. Check the condition of the electrical cord and plug. Check for coolant leakage at mounting points.
2. Check fuel warmer mounting, condition of hoses, and hose routing. Heaters with manual shut-off valves should be turned on before cold weather to ensure proper operation.
3. Check fuel/water separator mounting, condition of hoses, and hose routing. Drain any collected water.
4. Check the ether starting system by removing cylinder and activating the valve. Weigh cylinder for contents. Replace if low. “It is generally recommended to replace the starting fluid cylinder with a new one prior to each cold weather season.”
5. Oil pan heaters, battery warmers, etc.: Check element resistance with an ohm meter (300 watt-120 volt oil pan heaters have a normal resistance of about 50 ohms, battery warmers 75-125 ohms.) Check condition of electrical cord and plug.
6. Fuel-fired heaters normally are ready to fire up. Run the unit through at least two thermo-

statically controlled cycles. Check hoses for proper routing. Replace glow plug or igniter as required. Consult owner's manual for any other recommended tests. Tests should be done with the unit cold.

7. Electric Air Intake Heaters are OEM installed and are integrated with the engine ECM. Proper operation is often monitored by the ECM. Electric Air Intake Heaters require no scheduled maintenance or service.

General

1. Carefully check all engine and chassis hoses (fuel, oil, and air) for wear and deterioration. Replace as necessary.
2. Check operation of windshield washers and ensure that wipers remove solution without streaking.
3. Check windshield washer reservoir for proper type and amount of solution.
4. Check exhaust system for leaks i.e., black patches around connections.
5. Clean and grease electrical connections to the following: headlights, taillights, light cord receptacle, and clearance lights. Use approved dielectric grease.
6. Check fan blade and fan belt condition and tension.
7. Check fan clutch for play with the fan clutch released. Play should not be more than 3/16 inch at fan blade tip, maximum.
8. Lubricate door latches with a Teflon lubricant.
9. Lubricate door locks with dry graphite lubricant.
10. Check manual steering boxes for proper lubricant and level. Loosen bottom plate and drain out water.
11. Check transmission and differential oil levels. Check for milky color or other signs of water contamination.
12. Check fifth wheel adjustment. It should not exceed 1/8-inch play. Ensure sliders will move freely. Clean off excessive dirt and grease buildup.

Final Checks

1. With the engine at operating temperature and rated idle speed, check operation of the heater fan and check the temperature of air flowing out of the heater ducts. It should be 110°F (43°C) or more when ambient temperature is 30°F (-1°C) or higher and at 0°F (-18°C) ambient temperature, it should be above 90°F (32°C).
2. Road test the truck and check for leaks (coolant, oil, and air). The truck is now ready for cold weather operation. But there are a few extra things that may be required in your area. They can include, but are not limited to the following:
   - Chains that are in good condition and stored on the vehicle.
   - Emergency cold weather garments that are stored on-board.
   - Extra premixed antifreeze stored in the vehicle in a rugged container.

These are just three items mentioned by a majority of fleets. Your fleet may identify more.

NOTE: Please have the technician and shop foreman sign off on your maintenance log and date it.

ENGINE COOLANT

The cooling system is one of the most important parts of the engine that must be properly maintained to support cold weather operations. Maintenance must be scheduled in mid-fall prior to the onset of cold weather. The outline of the maintenance schedule should proceed in sequential order.

Inspection—A visual check of the entire cooling system (i.e., radiator, water pump, hoses, etc.). Look for leaks or dried inhibitor deposits (gel or crystals). Do the necessary repairs or adjustments (see "**Clean the Cooling System**" below).

Truck History—Have there been complaints from drivers about hot oil or water temperatures, lack of heat in cab, and/or chronic replacement of water or antifreeze? If so, proceed.

Clean the Cooling System—If inspection of the coolant reveals rust or turbid appearance or if deposits have formed on the top header of the radiator, clean the cooling system. Also clean the cooling system if truck history denotes the above mentioned chronic cooling system complaints. Then, follow the recommended cleaning procedure to remove rust, corrosion by-products, solder bloom, scale, and other deposits. Cleaning opens coolant passages and removes insulating scale and deposits that will cause overheating and possible engine block damage.

Antifreeze—Temperature protection to the lowest temperature expected should be the goal of any program. In most areas of the country, a 50/50 mixture, which protects to -34°F (-37°C), will provide necessary freeze protection.

Do not use more than 60 percent ethylene glycol. Excess antifreeze contributes to high concentration of total dissolved solids (TDS), loss of proper heat transfer, and is economically wasteful. Specify low silicate formulated antifreeze that meets American Society of Testing and Materials (ASTM) specification D4985.

Supplemental Coolant Additive (SCA)—Use of a SCA in conventional coolants is important to proper engine and cooling system protection. Follow manufacturer's recommendations.

If the system has been cleaned, a precharge of SCA must be added per manufacturer's recommendations and coolant volume. Otherwise, test coolant for proper additive level either by test strip or titration method.

Do the necessary adjustments to coolant if inhibitor levels are under/over the proper concentrations. Follow manufacturer's recommendations.

Reinhibit the cooling system with an SCA maintenance charge following the manufacturer's recommendations based on mileage or hours of operation. For further information, please refer to TMC RP 338, *Extended Service Interval Coolant*. Also see TMC RP 313C, *Checklist for Cooling System Maintenance*, and RP 302B, *Fleet Purchasing Specification for Ethylene Glycol-Base Anti-Freeze Coolant*.

ENGINE, TRANSMISSION, AND DIFFERENTIAL LUBRICANTS

As the temperature drops, lubricants become thicker and viscosity increases. SAE has addressed the viscosity grades for winter oils by using a "W" after the first weight number if cold temperature testing is required for qualification (10W, 15W-40, etc.). A 15W-40 engine oil will provide proper bearing lubrication when starting, down to an oil temperature of 0°F (-18°C). This is noted by observing the oil pressure indicator on the truck cab instrument panel.

When the temperature of the oil drops below 0°F (-18°C), there are two possible solutions to engine lubrication:

1. Anticipate the cold weather and use oil that has a better low temperature capability. This may include suitable synthetic oil.
2. Provide an auxiliary means of keeping the oil warm by use of oil pan heaters. It is recommended to "keep the oil warm" versus "getting the oil warm," as cold oil is hard to heat because higher viscosity does not allow great heat transfer.

If too thick, the lubricating oils in the transmission and differential will retard the engine starting RPM and make it more difficult to shift. When starting an engine, depress the clutch to reduce transmission drag. Light operation of the vehicle is recommended to ensure lubricants in the transmission and differentials have a chance to reach a fluid state. Use a transmission and differential lubricant that has a channel point lower than the lowest ambient temperature expected in your operating area. Thicker oil will require more horsepower to overcome internal friction, resulting in a substantially reduced fuel economy. The transmission and differential problems are best overcome by using a qualified synthetic lubricant.

It should be noted that petroleum-based oils are more susceptible to oxidation. This oxidation or degradation will result in thicker oil that will cause more difficulties in cold weather operation.

NOTE: Use a transmission and differential lubricant that has a channel point lower than the lowest ambient temperature expected in the operating area.

AIR STARTERS

In a low temperature environment, parasitic loads can negatively affect a diesel engine's startability. These same parasitic factors also will affect a starter's ability to crank. When these factors are properly addressed, their negative effects will be reduced.

The principle of an air starting system is the transfer of stored energy in the form of compressed air to a rotary motion through an air motor. A basic air start system is termed generically as "pre-engaged in nature." The following is a list of some cold weather operation precautions:

1. Air Reservoir: Dedicated solely to the air starter, divorced from its supply source with a check valve.
    a. Threaded joints and other possible leak points should be sealed properly with liquid Teflon sealant. This protects against corrosion.
    b. All steps must be taken to supply clean, dry air.
2. Ensure that the reservoir-mounted check valve lubricator, remote-mounted control valve, and air start relay valve supply lubrication at low temperatures.

3. Starter: Ensure proper lubricant is present at low temperatures.
4. Repair any leaks to ensure tanks remain fully charged. Also see TMC RP 126, *Air Starters.*

HOSES AND HOSE CLAMPS

The use of proper hoses and hose clamps are important for cold weather operation. Hoses should be inspected frequently looking for cracked, swollen, or dried out hoses. Squeeze the hoses to detect softness and fine-line cracks. Clamps should be tightened firmly and retightened at maintenance intervals.

When using silicone hose, use silicone hose clamps and tighten to manufacturers specification. If the hose is a tight fit when installing, lubricate with water and/or liquid dish soap. Do not lubricate with ethylene glycol and do not use wire-type hose clamps.

There are spring loaded constant torque hose clamps available which help compensate for expansion and contraction. Please refer to the clamp manufacturer's torque requirements as they vary based on clamp design. Also see TMC RP 303A, *Silicone Hoses and Hose Assemblies*, TMC RP 313C, *Checklist for Cooling System Maintenance*, and RP 332A, *Guidelines for Hoses, Clamps and Fittings for Cooling and Charge Air Cooler Systems*.

BATTERIES

Consult TMC RP 109A, *Battery Ratings and Engine Cranking Requirements*, as it ties the engine requirements, oil viscosities, and minimum battery ratings together. Also, TMC RP 105A, *Battery Cable Assemblies*, should be used to ensure cable resistance values are not exceeded. TMC RP 125, *Battery Vibration Standards*, that covers improved battery life and reducing the rate of capacity loss during use by using approved mounting practices.

For cold weather operation, cold cranking amps (CCAs) are very important. You must know the CCAs the batteries deliver at the ambient condition you operate in between the range of 0 to -20°F (-18°C to -29°C). The reserve capacity rating (RCR) is important if long-term parking with electrical accessories or lights turned on are common to your operation. See TMC RP 109A, *Battery Ratings and Engine Cranking Requirements*.

BATTERY WARMERS

As a battery gets colder, its available energy (cranking power) decreases. While a battery at 80°F (27°C) has 100 percent of its power, that same battery at 14°F (-10°C) has only 50 percent power available. When the temperature drops to -31°F (-35°C), only 10 percent of the original power still is available. Also, as the temperature of the engine decreases, it becomes more difficult for the starter to crank it over. An engine at -20°F (-30C) is 3-1/2 times more difficult to crank as an engine at 80°F (27°C).

There are various styles of battery heaters available, the most common of which are the blanket type and the plate type. The plate type battery heater usually is placed under the battery, but it can also be placed between batteries. The blanket type of heater comes in a variety of lengths and wattages. It also incorporates its own insulation.

It is best to heat a battery slowly. Also, a well-insulated battery box helps not only when heating the battery, but helps maintain heat.

DIESEL FUEL

In preparing for cold weather operations, fuel plays the most important role. There are three areas relating to fuel that should be examined. They are fuel storage, vehicle configuration, and low temperature fuel operability. The following are suggested guidelines and related terms used when dealing with fuel and fuel systems winterization.

Fuel Storage—Underground storage tanks can form condensation. Checks for water need to be diligent and any water found in the system must be removed.

Positive water removal filters need to be installed on fuel dispensing pumps.

Extra concern should be given to microbiological contamination. These living organisms can wreak havoc at any time, but they can really cause problems in the winter due to the water they can generate.

Tank fill areas should be raised to eliminate the possibility of water, snow, or debris from entering through these areas. Tank vents should be checked to ensure only air could enter them.

Vehicle Configuration—Vehicle fuel tanks should be checked for water contamination and any water or debris removed if found. Crossover lines should be checked for both correct size and low spots that can collect water. Any exposed fuel lines, especially in low areas can cause freeze-ups if water is present during severe cold temperatures.

Fuel tank vents must be kept free of frost or ice for the proper functioning of a "dual action, dual return" system.

Various types of fuel heaters are available and should be selected based on vehicle geographic location, geographic location of traveled areas, and type of fuel that is expected to be used.

Fuel Operability—Considering winter fuel is produced by refineries on a regional basis, it should be recognized that all fuel is not the same. ASTM D975 specifications for winter fuel are based on a 10 percent mean low temperature for that month and region only. As an example, regionally adjusted fuel in Atlanta will have a cloud point significantly different than a fuel in Minneapolis. For more information on minimum fuel specifications in your region, please refer to TMC RP 304B.

FUEL WARMERS

For the broad temperatures involved in this RP—ranges of +40 to 0°F, 0 to -20°F, and -20 to -40°F (+4.5 to -18°C, -18 to -29°C, and -29 to -40°C)—it recognize that most engine fuel starvation occurs as wax crystals form when the temperature of the diesel fuel in the suction line falls below the cloud point where wax crystals form. Depending on the chemical composition of the fuel, the exact solidification rate and operability in individual fuel systems will vary.

Wax and ice crystals impede fuel flow through filters, and in some instances tank suction line elbows. Unfortunately, the cloud point of No. 2 diesel fuel is not publicized and varies widely by supplier, cost, quality, and geographic area. Also, the temperature of the fuel in the suction line at a given ambient temperature may vary widely between different engine makes due to the differences in the volume of warm fuel returned to the fuel tank.

The fleet operator has three options that can be used to lower the cloud point or improve the operability of fuel. The first is to blend with kerosene, the second option is to employ cold flow improving additives and the third option is idling the truck to return hot fuel. Idling is not recommended; please check your state idling regulations. See "**FUEL HEATERS**" and "**AUXILIARY FUEL-FIRED CAB AND ENGINE HEATERS**" within this RP.

The two most practical heat sources for fuel heating during highway operation are electricity and engine coolant. However, for various reasons, coolant-type heaters dominate Class 7 and 8 commercial vehicles. The electrical system capacity of trucks limits electric heater outputs.

A disadvantage of coolant type fuel heaters lies in the fact that a free standing truck that has been cold soaked at low temperatures can have a fuel filter wax-up after start-up and before a sufficient coolant temperature increase can be reached. This can be overcome by using an on-board 12-volt filter preheater, by keeping the coolant warm with fuel fired coolant heaters, or by using electrical block heaters.

The placement of the fuel-heating device can be critical and is usually partly based on heater design. The nearer it is mounted to the fuel filter, the less heat rise is required. When mounted back near the fuel tank, a greater heat rise is required to overcome the heat loss through the fuel line leading to the fuel filter. Fuel heaters mounted in the tank are regarded as an explosion hazard and are not recommended.

The final problem for consideration is paraffin wax or ice crystals that may form at the fuel tank top or bottom suction elbow. This can be solved by the use of a 12-volt heating device thermostatically or manually controlled. See SAE J1422, "Fuel Warmer-Diesel Engines."

FUEL HEATERS

Fuel Heater Selection

Selecting the proper size fuel heater, correct location, piping and controls to ensure the maximum efficiency are important consideration for heavy-duty diesel applications. Before deciding on a fuel heater, check each engine to find the fuel flow rate. Do not confuse fuel consumption with fuel flow. In most cases, fuel flow is greater than fuel consumption and could be as much as three-to-four times greater than fuel burned, with the unused fuel being returned to the tank. This unused fuel is used as a coolant to carry excessive heat away from the injectors. Fuel flow rates can be anywhere from eight gallons per hour (GPH) to well over 150 GPH.

Electronically controlled engines have maximum allowable fuel temperature requirements. The engine/truck manufacturer can supply data on these temperatures. The fuel heater or an auxiliary device to the fuel heater should have an automatic capability to limit the fuel temperature to meet these requirements.

A fuel heater designed to raise the fuel temperature approximately 60°F (16°C) at 125 GPH would undoubtedly be too large for a 40 GPH fuel system. One designed for 40 GPH would be too small and create too great a pressure drop for a 120 GPH system.

It is important to check pump model numbers, as the same engine manufacturer may produce the same type and style engines with pumps of widely different capacities. It is equally important to know the maximum pressure drop allowed at the fuel transfer pump inlet and the pressure drop on your unit. The truck manufacturer can supply these specifications. If the truck has had another filter, water separator, longer lines, or smaller diameter lines added to the fuel system since it was new, it is possible that the pressure drop has increased above the maximum allowable limits. Therefore, adding a smaller capacity fuel heater may further overload the seals on the transfer pump, causing premature failure.

Next, check the location of the primary filter, the distance between primary and secondary filters, and if considering the coolant or "tank type" fuel heater, the accessibility of water lines and available space for mounting.

Fuel heaters should be mounted as close as possible on the inlet side of the primary filter. The longer the lines, the greater the pressure drop and heat loss. Temperature difference could be as much as 30°F (-1°C) due to heat loss from the fuel heater to the injection pump.

In summary, the following information is needed:

1. Fuel flow rate in GPH or gallons per minute (GPM).
2. Maximum allowable fuel temperature of the engine.
3. Maximum allowable pressure drop.
4. Present pressure drop on your system.
5. Available space for mounting fuel heater.
6. Anticipated ambient temperature and maximum temperature rise required.

With this information, you may select a fuel heater.

What to Look For in a Fuel Heater

Consider the following when selecting a fuel heater:

1. Construction and materials.
2. Sizes of openings for both fuel and water. Remember, you will be adding more fuel lines and possibly 90-degree fittings. The pressure drop is so great in a 90-degree elbow that plumbing engineers indicate, "nine-90 degree elbows equal one shut off valve." Therefore, do not use 90-degree fittings. If 90-degree fittings cannot be avoided, go to the next size larger fittings (i.e., 1/2 inch N.P.T. go to 3/4 inch N.P.T.).
3. Pressure drop of the fuel at the specified GPH or GPM of the engine. Add this figure to the present pressure drop to see if it would still be within allowable limits.
4. Fuel temperature differential at inlet and outlet of fuel heater, at engine's maximum fuel flow rate, with water at 180°F (82°C). If the water used is from the return line of the cab heater, then 160°F (71°C) water temperature should be used.
5. In the space available, can the unit be mounted as the manufacturer specifies? The above applies to the coolant, exhaust, and mixer type fuel heaters (mixes return fuel with cold fuel to help prevent waxing). Water temperature has no effect on the two latter types.

Fuel Heater Installation

Try to mount the fuel heater in a position that will allow the coolant to drain out when the engine is drained. Once the coolant type fuel heater is mounted, it is wise to connect the inlet (cold fuel line) opposite from where the coolant line enters the fuel heater. This will cause the fuel to flow against or into the hotter coolant that could result in a slightly higher fuel temperature.

Engine and truck manufacturers recommend the use of cab heater lines for the hot coolant. This avoids connecting or disconnecting lines in the spring and fall. Some trucks have an automatic water shut-off controlled by the cab heater controls, while others have a manual shut-off cock on the engine.

Always install the fuel heater on the inlet cab heater hose, not on the return line. In either case, when the cab heater is shut off, the fuel heater will not function. If the cab heater lines are not used, be sure to install a shut-off cock in the water line.

It is very important that all fuel heaters be installed between the fuel tank and the primary filter and always lower than the fuel transfer pump, to avoid a possible point of air entrapment.

The fuel line from the fuel heater to the primary filter should be a fuel hose type with a rubber interior, cotton braid, and steel outer braid to act as an insulator in keeping the heat in the fuel. In cases where the fuel heater's capacity is marginal, it is necessary to insulate all lines between the fuel heater and up to and including the fuel filters, in order to retain as much heat as possible. In some cases, the fuel could be just above the cloud point when going through the primary filter and below the cloud point as it reaches the secondary filter. This could lead to paraffin wax plugging the secondary filter and eventual fuel starvation.

With coolant type and exhaust type fuel heaters, all return fuel goes directly back to the fuel tank.

Most engine manufacturers desire a fuel temperature of 90-100°F (32 to 38°C) and will accept fuel temperatures up to 160°F (71°C) without detrimental effects to the engine or injection system. However, fuel temperatures over 100°F (38°C) usually results in a loss of power. Roughly one percent of engine horsepower is lost for every 10 degrees of fuel temperature over 100°F (38°C). Thus, if fuel temperatures were operating at 160°F (71°C), the engine would be derated by about six percent of its maximum power at full throttle.

Coolant type fuel heaters without thermostatic controls limit the fuel temperature by the coolant temperature, heat-exchanging efficiency of the unit, and the heat loss between the fuel heater and the injection pump. On stationary and marine engines where the return fuel raises the fuel tank temperature to over 200°F (93°C), the fuel heater acts as a cooler to bring fuel temperature down to a more acceptable level.

Coolant type fuel heaters can also be used with engine oil. Any leakage of these units would be into the fuel, as the fuel system is under negative pressure while the water or engine oil could be 50 psi or greater.

FUEL/WATER SEPARATORS

A TMC Cooling Systems Task Force survey indicated more than 70 percent of the responding fleets use fuel/water separators. There are many types and sizes available on the market today. You must know the required fuel flow rate, usually expressed in gallons per hour, which will need to flow through the fuel/water separator. There are three basic types to choose from: Regular fine mesh filters, coalescer screens, and fuel/water separators containing no filters or screens.

When using fuel/water separators during cold weather operation, you must consider the following:

1. Mount the fuel/water separator in a warm protected area, usually under the hood. Some fleets will wrap them with insulation for cold weather operation.
2. If a diesel fuel warmer is used in conjunction with a fuel/water separator, mount the separator down stream from the fuel warmer so it receives warm fuel.
3. Do not use 90-degree elbow fittings; these act as freeze points in cold weather.
4. Drain the fuel/water separator frequently so there is minimal water in the separator bowl. This water turns to ice after shutdown and takes a long time to melt at start-up. Ice crystals can plug a fuel filter just like wax crystals.
5. See TMC RP 317, *Fuel/Water Separating Devices.*

OIL PAN HEATERS

In cold climates, a coolant-type engine heater may not be sufficient to ensure engine startability. Oil pan heaters can be used to assist other types of cold weather starting aids. If the engine is air cooled, an oil pan heater is a very effective way to warm the engine before starting.

Cold oil increases the load on the starter. Starting an engine with unheated oil greatly increases the wear on the engine and shortens the useful life of the oil. Using an oil heater can minimize these problems. Always use the heater to keep the oil warm, not to raise the oil temperature after a cold soak.

When using an oil pan heater in the higher watts/quart range, a heater thermostat is recommended. Upper limits for these thermostats are typically 120°F to 170°F (49°C to 77°C).

There are two basic types of oil pan heaters available, clamp-on and immersion types. The clamp-on type heats the outside of the oil pan that transfers the heat to the oil. One should know the construction detail of the oil pan, as some engine manufacturers use a plastic pan liner. When using an immersion type, 8-10 watts per quart of oil in the sump is sufficient to maintain the oil 80°F (27°C) over ambient temperature. When higher oil temperatures are required, do not exceed 20 watts per quart of oil and use a thermostat-equipped heater.

As with all electrical devices, the power cord should be routed and secured to avoid damage. Reference: SAE J1310, “Electric Engine Preheaters and Battery Warmers for Diesel Engines.”

COOLANT HEATERS

Coolant heaters are used to aid engine starting and reduce engine wear caused by cold starting. There are two basic types of heaters used today, circulating tank style and immersion block heaters. Another type mentioned by some fleets, fuel fired heaters, is covered elsewhere in this Recommended Practice. The immersion type of coolant heater is installed directly into the engine block, in a location normally provided by the engine manufacturer. The heated coolant circulates around the cylinder walls by convection.

The most common immersion heaters are powered by 120 or 240 volts AC and are available in wattages up to 2500 watts. A simple rule of thumb is to use 1.5 to 2.5 watts per cubic inch of engine displacement. To determine heater element wattage, measure the heating element resistance and compute the heater wattage by using the known nominal heating element voltage and following formula:

$$\text{Watts} = \frac{(\text{Volts})\ 2}{\text{Ohms}}$$

The circulating tank-type heater sets up circulation through the engine by the way it is installed. The coolant leaves the bottom of the block and travels to the heater. The heated coolant rises and is plumbed back into the top of the block. Generally, tank-type heaters should be of higher wattage than immersion heaters because some heat is lost from the hoses and tank body itself. Circulating tank-type heaters are most commonly available in 120 or 240-volt AC versions and are available in wattages up to 3500 watts. There are also non-electric models available and they are usually propane fired.

Coolant heaters should not heat the radiator, heater core, and any long hoses. They all dissipate heat.

All of the above heaters are available with energy-saving thermostats, but normally they are not sold with thermostats so you should ask your supplier for the correct thermostat, etc. for your application.

A heater must be sized to provide the required engine temperature for the environment in which the vehicle is operated. For diesel engines, an average engine temperature of at least 30°F (-1°C) is recommended. To aid your selection, please see the cold weather operating aids chart at the end of this RP.

The coolant system is critical to engine heater operation. A proper antifreeze and water mixture (50/50, max. 60/40) must be maintained. The heating element produces a hot spot in the engine and an over concentration of antifreeze will allow silicates to drop out, coating the heating element. This will cause the heater to fail prematurely. In severe cases, this can occur in a few hours of heater operation.

A typical engine heater requires minimal maintenance, but should be inspected prior to the high use season. This is covered under the section entitled **Cold Weather Operation Maintenance Checklist** in this RP as part of the general maintenance program. See SAE J1310, “Engine Pre-heaters for Diesel Engines.”

ENGINE STARTING FLUID (ETHER) STARTING SYSTEMS

⚠CAUTION: The use of aerosol spray cans of Engine Starting Fluid (ether) is not recommended. This section deals with and is describing Engine Staring Fluid Systems that are permanently installed on the vehicle either by the vehicle OEM or as retrofit kits.

⚠CAUTION: Engine starting fluid injection systems (Ether) should not be used on in-direct injection (IDI) “glow plug” diesel engines or on direct-injection diesel engines that have glow plugs installed directly in the combustion chamber. Engine starting fluid injection systems are used primarily on direct-injection (DI) diesel engines that **do not** have glow plugs in the combustion chamber. If the engine is equipped with glow plugs and is still difficult to start the glow plug system may need to be serviced.

Many direct injection diesel engines provide electrical grid heaters as a starting aid. Some engine manufacturers warrant the use of an engine starting fluid system in conjunction with the electrical grid heater. The engine manufacturer, service manual, or ether system supplier can be consulted regarding the use of ether with an electrical grid heater.

Background—Diesel fuel has an auto-ignition point of approximately 725ºF (385ºC). A diesel engine relies on the heat generated by the compression stroke to ignite the diesel fuel. At some point, as the ambient and engine temperature’s decline, the heat generated by the compression stroke will be insufficient to ignite the diesel fuel; the engine will not start or will become very difficult to start in a reasonable amount of cranking time.

Engine starting fluid (a.k.a. ether) has an auto-ignition point of approximately 350ºF (177ºC). When ether is injected into an engine's air intake system, in the proper ether to air ratios, the engine starting fluid mixture will ignite, initiate the combustion process, and allow the injected diesel fuel to ignite and burn in a very efficient manner. The optimum concentration of ether versus air is 3.37 percent by volume. This ratio creates the most effective and efficient burning velocity needed to achieve a successful cold engine start.

Engine starting fluid systems can vary from the hand held aerosol spray can (totally under the influence of the operator) to the fully automatic, electronically controlled (Engine ECM) system that is wired directly into the engine's electronic control module (no operator influence). There are also many operator controlled, push button (also known as measured shot type) systems available.

About 12 CC's of starting fluid per second are typically dispensed with an aerosol spray can. An operator normally will put in at least two to three seconds of spray or 35 CC's — six times the amount of ether that is required. This improper/excessive usage of ether has created the "ether is bad" image. Improper/excessive use of ether can cause "pinging", ether lock up, and possible engine damage. As previously stated, the use of the aerosol spray can is not recommended.

With a measured shot-type system the operator pushes and releases a push button switch (typically located on the dashboard of the vehicle), which will then release a "measured shot" of ether into the engine. On a measured shot type system, the measured shot valve size will determine the overall quantity, per shot, that the system will deliver and the atomizer/nozzle will determine the rate or flow at which the "shot" is injected into the engine. Measured shot valves are available in four different "shot" sizes. Normally for a Class 6, 7 or 8 diesel truck engine, the "6 cc" shot size is usually recommended. The atomizer/nozzle that is selected will be dependent upon the actual engine being outfitted. The engine manufacturer and ether system supplier's recommendation for the proper atomizer/nozzle selection should be followed. On any measured shot system, a thermal switch device can be wired in series with the ether valve to prevent ether from being injected into a warm engine (another common abuse of ether) by the operator.

A push-pull cable controlled measured shot system is available and operates in the same manner as the push button measured shot type system; pulling the cable (instead of pushing a button) charges the shot chamber and pushing the cable (instead of releasing a button) dispenses the ether to the engine.

An automatic starting fluid system is available either as a stand alone-independent-engine temperature controlled type, independent electronically controlled type, or an ECM (engine control module) controlled type system. Ordinarily an automatic starting fluid system will use an "activated flow" valve, not a measured shot valve. An activated flow valve will deliver the engine starting fluid whenever the valve is activated or energized. Typically an automatic system will also use an internal valve regulating metering orifice to control the rate or flow of starting fluid. Sometimes the automatic type system will not rely on the atomizer/nozzle for regulating the flow rate. The service manual or the system manufacturer can be consulted for proper identification of the valve flow mechanism.

On both the measured shot and the automatic type systems, the atomizer/nozzle is the device that directs the starting fluid into the airstream of the engine. The recommended location for the atomizer/nozzle is in the engine intake air system, upstream of and as close to, the inlet of the intake manifold as possible. The direction of spray should be towards the incoming airstream in order to obtain maximum atomization. Nylon or metal tubing generally connects the valve and atomizer/nozzle (Reference: Information Report SAE J2079, "Location of Atomizer of Ether Systems").

The most basic automatic starting fluid system (the stand alone-independent engine temperature controlled type) is simply wired in series with the engine cranking motor circuit, through an engine temperature sensor. The engine temperature sensor will allow the ether valve to activate and inject starting fluid while the starter is engaged if the engine is cold enough to warrant the use of starting fluid.

Typically an electronically controlled (either independently controlled or controlled by an engine's electronic control module or ECM) starting fluid system will monitor basic engine requirements in order to warrant or enable the injection of starting fluid. The electronically controlled system is able to monitor such items as engine speed, engine temperature, engine start-run-or cranking disposition and overall time of the ether injection event.

An independent electronically controlled type starting fluid system uses a separate electronic control module (other than the engine's own ECM) to enable the injection of starting fluid when needed.

On an ECM driven starting fluid system all the parameters and controls regarding the use of ether reside within the engines own ECM. Virtually all domestically produced, heavy-duty diesel engines have the required "ether system control logic" embedded in their ECM's; the engine manual or dealer can be consulted for availability. Sometimes an interface or relay may be required between the ether solenoid valve and the engine's ECM if a direct connection is unavailable. Some of the ECM's provided by an engine manufacturer will provide for a "Sourcing Output' or battery + positive connection for the ether solenoid valve and other engine manufacturers ECM's provide a "Sinking Output" or battery ground connection. The engine service manual should be checked for proper identification.

Many starting fluid system can be retrofitted. There are some generations of ECM's which will not / cannot accept or be upgraded to ECM driven type systems. Check with your engine service manual for compatibility and applicability.

AIR BRAKES

All air brake-equipped vehicles rely on clean, dry air for proper and consistent operation of their air brake systems. This includes the air compressor, reservoirs, brake valves, and accessory items such as horns, wipers, and leveling valves on air suspensions.

Air in the brake system can include contaminates such as road dirt, salt, oil, and various other pollutants. It is most important to remove these contaminates, particularly during cold weather operations.

There are several methods to protect the air brake system. Some are:

1. Alcohol injectors. They release vaporized methyl alcohol into the system to prevent freezing.
2. Automatic moisture ejectors mounted at the reservoir operated via governor or foot valve application, or manual type operated by a cable.
3. Air dryers mounted in line between the compressor and air reservoir.

There are many various types of these units, including a desiccant type that uses special pellets to absorb water and trap pollutants. Several air brake manufacturers offer this type to the fleet market and over the years, modifications and enhancements have been made to improve the efficiency of this type. Maintenance of desiccant types is required.

The aftercooler type also mounts in line between the compressor and supply reservoir. This model removes the same containments. As air enters the aftercooler and condenses, pollutants and moisture are removed. This type of dryer relies on placement of the unit so there is proper airflow around the unit to maximize its cooling capabilities.

ELECTRIC AIR INTAKE HEATERS

Electric air intake heaters utilize an electrical resistance-heating element integrated into the intake air system. Typical wattages are 1.0-3.0 kilowatts at 12 or 24 volts.

Electric air intake heaters are used to warm the intake air before it enters the cylinders. The increase in cylinder temperature will improve combustion during a cold start and initial engine warm-up. The improved combustion provides a smoother and quicker start-to-idle operation. The reduction of unburned fuel provides the benefit of reduced white smoke and other customer discomforts. The engine ECM usually controls air intake heaters. No scheduled maintenance is usually required.

NOTE: Additional battery capacity maybe required for successful starts when electric grid air inlet heaters are used.

FUEL FIRED INTAKE MANIFOLD HEATERS

Fuel-fired intake manifold heaters warm the intake air by spraying diesel fuel into the manifold and igniting the fuel with a small electric heater. The heated gases then flow into the cylinders to improve combustion during a cold start.

AUXILIARY FUEL-FIRED CAB AND ENGINE HEATERS

Fuel-fired heaters are gaining popularity in the U.S. market. They have been used in European countries for a number of years. They are used to heat the engine and cab/sleeper via the coolant by burning fuel, usually diesel fuel No. 1, No. 2, or a blend of both. There are also gasoline-fired versions.

For Class 6, 7, and 8 trucks, one should use a heater rated about 20,000 to 25,000 British Thermal Units (BTU) per hour (btu/hour). This amount of heat also will heat a sleeper cab. If you are equipping a Class 8 vehicle with a double-60 inch or larger sleeper cab for -40°F (-40°C) use, then one should use a 25,000-40,000 BTU/hour rated heater.

Always route the coolant from the heater to the cab/sleeper, then to the engine, and finally from the engine back to the heater. Some people also opt to route coolant from the engine to an in-tank fuel warmer and then back to the heater. This way, the fuel gets pre-heated also.

During summer, it is important to operate the heater 15 minutes per month to keep the system clean. Some engine manufacturers recommend the coolant heater lines be shut off when operating the vehicle in warm ambient temperatures.

SHUTTERS AND WINTERFRONTS

Radiator shutters were used in conjunction with the engine thermostat to maintain proper engine operating temperatures. In some rare cases, they are still used in some applications. Radiator shutters regulate airflow through the radiator core and, therefore, are of particular benefit in cold ambients for maintaining engine coolant temperature and providing improved cab heating. By improving engine operating temperatures, shutters also help to combat reduced fuel economy brought on by the cold environments referenced in this RP.

When using radiator shutters, ensure that the controls are properly coordinated with fan clutch actuation and any override controls that may be needed , as dictated by air conditioning or intake air temperature (as in the case of air-to-air charge air cooled diesels). See TMC RP 316A, *Engine Temperature Control Settings for Liquid Cooled Diesel Engines*, for proper control setting recommendations.

Winterfronts can also be helpful in cold climates by reducing the amount of ram air circulating through the engine compartment. Refer to engine or vehicle manufacturer's recommendations on use of winterfronts and for determining which application is appropriate for ambient conditions. See TMC RP 343A, *Charge Air Cooler Winterfront Application and Design Guidelines*.

TABLE 1
COLD WEATHER OPERATING AIDS

Temperature	Aftercooler	Ether Starting Aid	Coolant Heater	Oil Heater	Under-hood Air	Fuel Heater	Battery Heater	Radiator Shutters	Engine Enclosure	Winter Front	Thermatic Fan
50 to 32°F 10 to 0°F	Recommended	* Required									
32 to -10°F 0 to -23°C			Required		Required	Recomm.	Required	Required			
-10 to -25°F -23 to -32°C				** Required		** Required			Required	Required	Required
-25 to -65°F -32 to -54°C											

Courtesy of Cummins Engine Company
* Required dependent upon engine model.
* * Required dependent upon viscosity / pour point.

NOTE THE FOLLOWING:

1. Generally, for all starting aids, the size of the device is dependent on engine size. The larger the engine, the larger the device must be. Your supplier can help you size the unit.
2. Generally, when your operating environment approaches 0°F (-18°C), more than one type of starting aid is required to successfully start and operate the vehicle.
3. When your operating environment approached -25°F (-32°C), several types of starting aids are necessary.
4. Aftercoolers are used on all of today's medium and heavy-duty on-highway diesel engines. The aftercooler is essentially an air-to-air heat exchanger that cools the turbocharged compressed hot intake air. The device is located before the engine intake manifold either in front of, parallel to or in back of the coolant radiator. Aftercoolers are necessary to optimize the fueling characteristics of the diesel engine with minimal exhaust emissions.
5. The thermatic fan drive is a demand-only feature to minimize parasitic losses when not required and very beneficial during engine warm up time, not forcing air across the aftercooler package. The thermatic fan drive can be engaged due to excessive engine / fluid temperature, transmission temperature (if the transmission cooler package is mounted in the cooling fan stream) or from the climate control air conditioning system requiring air flow across the condenser core, generally located in the cooling fan stream. The thermatic fan drive improves fuel efficiency of the vehicle when disengaged.
6. This RP does not cover cold weather operation below -40°F (-40°C). For temperatures below this rating, please consult your truck and engine manufacturer.

TMC Recommended Practice

Proposed RP 356 (T) **VMRS 044**

COLD FLOW OPERABILITY OF DIESEL FUEL

TABLE OF CONTENTS

PURPOSE AND SCOPE .... RP 356-1
FUEL GELLING .... RP 356-1
PARAFFIN WAX SOLIDIFICATION, CRYSTAL SIZE AND FUEL FILTERS .... RP 356-1
WINTER FUEL CHALLENGES .... RP 356-3
FLEET PRACTICES TO AVOID FUEL GELLING .... RP 356-3
ULTRA LOW SULFUR DIESEL (ULSD) FUEL COLD FLOW OPERABILITY .... RP 356-3
BIODIESEL .... RP 356-4
CONTAMINANTS THAT MAY IMPACT FUEL COLD FLOW .... RP 356-4
FUEL COLD FLOW OPERABILITY RECOMMENDATIONS .... RP 356-5
PETROLEUM INDUSTRY COLD FLOW TESTING PRACTICES .... RP 356-5
REFERENCED DOCUMENTS .... RP 356-7

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to offer suggestions on cold flow operability solutions based upon available test methods and fleet fueling practices. This RP provides a basic understanding of petroleum refining, cold property testing methods employed and the fuel delivery infrastructure as it relates to fleets' operational requirements.

This RP addresses both diesel fuel cold flow operability problems associated with ambient temperatures below 32°F (0°C) and biodiesel fuel use. It also covers available testing methods for fleets to perform, or subcontract, cold flow operability tests.

FUEL GELLING

Fuel gelling, technically referred to as clouding or cloud point (CP), is of special concern to truck fleets that travel through cold and diverse climates within North America. Clouding may be described as the point when naturally occurring diesel paraffin comes out of solution forming wax crystals. It is these wax crystals that plug fuel filters and create undesirable down time and costly delays.

PARAFFIN WAX SOLIDIFICATION, CRYSTAL SIZE AND FUEL FILTERS

Figures 1-3 illustrate the rapid solidification rate of wax crystals when a fuel is maintained at cloud point for up to two minutes. The CP temperature was obtained according to ASTM D5773[1].

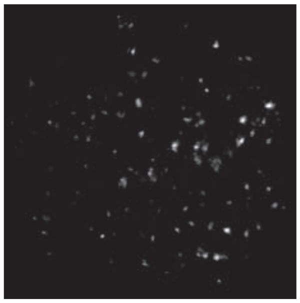


Figure 1: Fuel at CP

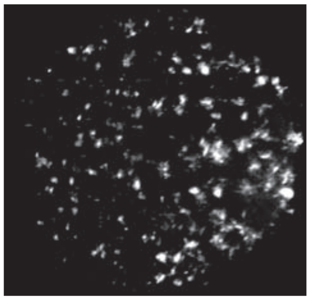


Figure 2: Fuel Maintained at CP for One Minute

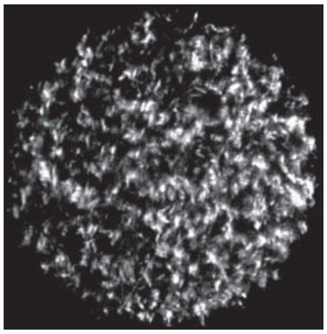


Figure 3: Fuel Maintained at CP for Two Minutes

Figures 4-9 illustrate the micron (μm) size of wax crystals of three B100 biodiesel fuels, two petroleum distillates fuels and one biodiesel blend. Fuels were stabilized at CP for two minutes according to ASTM D5773. (Microscopy photos courtesy of Phase Technology.)

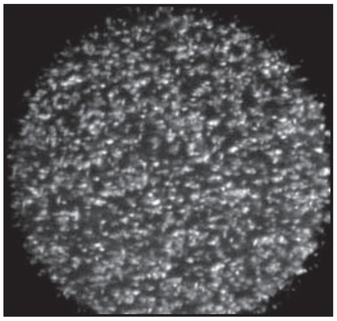


Figure 4
Soy B100, ~8 to 13μm
CP = 34.3°F (+1.3°C)

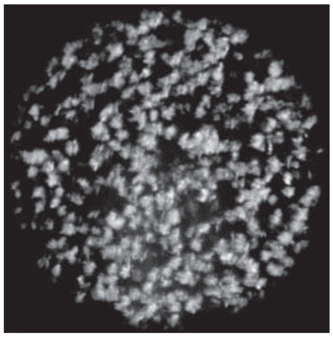


Figure 5
Yellow Grease, ~35μm
CP = 37.4°F (3°C)

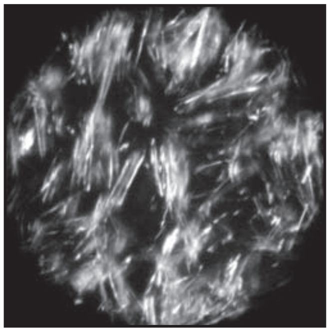


Figure 6
ASTM RR1, ~150μm
CP = 31.8°F (-0.1°C)

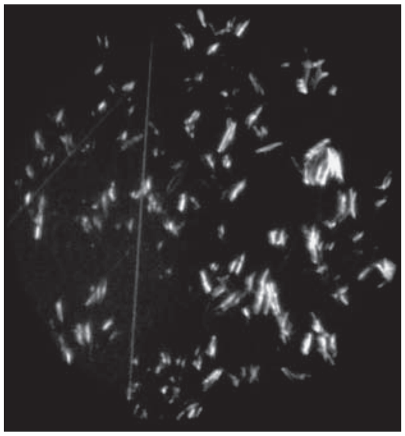


Figure 7
ASTM DL0110, ~30μm
CP = 17.1°F (-8.3°C)

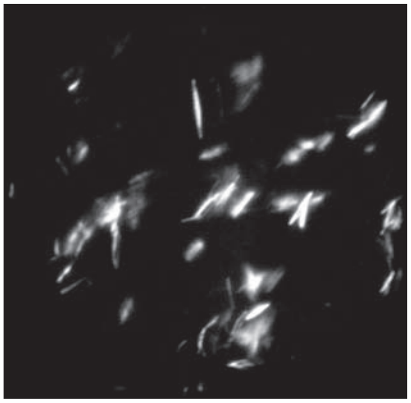


Figure 8
ASTM DL0010, ~70μm
CP = -2.0°F (-18.9°C)

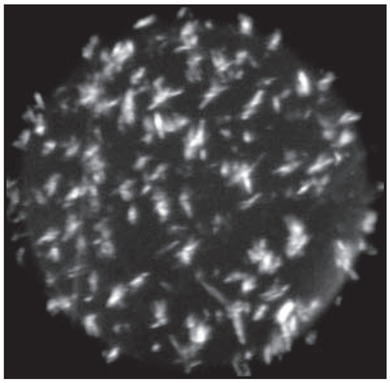


Figure 9:
Biodiesel (B20), ~30μm
CP = 19°F (-7.2°C)
80% DL0110 & 20% ASTM RR1

Today's diesel trucks have fuel filters with typical ratings ranging from 2µm to 15µm (micron). For untreated fuels, gelling typically occurs at or slightly below CP. When considering the use of cold flow improving additives, consult the additive supplier for recommended dosing rates and anticipated fuel cold flow improvement. It is important to know that no two fuels are identical—cold flow improvements will vary depending on the specific interaction between the chemical additive and the fuel blend. Considering chemical interaction cannot be predicted in a precise manner, the most effective way to determine improvement in cold flow behavior is to perform fuel tests using accepted industry methods. Details regarding test methods appear in the section **Petroleum Industry Cold Flow Testing Practices**.

WINTER FUEL CHALLENGES

In the wintertime, petroleum refiners' supply seasonally adjusted diesel fuel to various regions of the United States and Canada; fleets often refer to these fuels as winter blends. Due to diverse climatic requirements, there are well over 100 temperature zones to satisfy. Even within one U.S. state, there exist multiple zones that are very different in ambient temperature and cold flow operability requirements. Typically, petroleum refiners produce fuels that are 6°C colder than the tenth percentile minimum ambient temperatures. These temperatures are historical average lows from October to March as noted in the American Society for Testing and Materials (ASTM) standard D975[2].

From a fleet operability perspective, fuel purchased in Florida will have a significant difference than fuel purchased in Minnesota. Using the month of December as an example, fuel purchased in Tampa, Florida will have an approximate CP of 18°F (-8°C), which is significantly different than fuel purchased in Minneapolis with a CP of approximately -33°F (-36°C)—so much so that the fuel may be inoperable in the northern climate.

Compounded with the fact that local cold snaps are unpredictable, it's not practical for refiners to supply hundreds of winter blends to satisfy the needs of every locale at all times. In the event of a cold snap, refiners may take two weeks or more to deliver adjusted product to market. For more information on historical average low temperatures in your region, please refer to TMC RP 304B Minimum Specifications for On-Highway Diesel Fuel Type No. 2-D [3] or ASTM D975.

FLEET PRACTICES TO AVOID FUEL GELLING

Uncertainty over fuel quality has prompted drivers to employ practices that result in additional expense, productivity loss and unnecessary anxiety. The most common practices that truck drivers use to avoid fuel gelling include:

- frequent refueling along their route in order to purchase seasonably adjusted fuel,
- the use of fuel heating devices,
- excessive idling in fear that the truck will not restart, and;
- blindly diluting their fuel with additional No. 1 fuel (kerosene) or chemical additive.

⚠ CAUTION: Excessive idling will result in higher operating costs. Fleets are also advised to comply with all applicable anti-idling regulations.

Fortunately, equipment users and truckstops can improve the winter quality of their diesels by blending a lighter diesel stock, commonly referred to as No. 1 fuel (kerosene) with the regular No. 2 diesel or winter blend. They may also achieve positive results with the use of chemical additives. In either case, equipment users can save money and avoid potential downtime by knowing the CP of the fuel before blending or additive treatment. It is extremely important to know the CP of your fuel before any contemplated action. For a more detailed explanation of CP and the limitations of both the cold filter plugging point (CFPP) test and the low temperature flow test (LTFT), see section **Petroleum Industry Cold Flow Testing Practices**. For information on cold weather operations involving ancillary heating devices, see TMC RP 311, *Cold Weather Operation*.[4]

ULTRA LOW SULFUR DIESEL (ULSD) FUEL COLD FLOW OPERABILITY

During the manufacture of ULSD, additional cold flow problems may arise as a result of chemicals changes during the desulphurization process. In order to monitor changes in ULSD cold flow characteristics, many petroleum refiners began to install online process control systems. The process control systems carefully monitor the CP of fuels, in real-time, and adjust the blending process which produces fuels very close to minimum specifications as defined by the customer.

Fuels are released into markets that commonly exceed the tenth percentile minimum ambient temperatures within each geographic region as defined in ASTM D975. In the refinery industry, exceeding minimum specifications is referred to as "product giveaway", an action that is generally unfavorable to the profitability of refineries. "Product giveaway" or a fuel CP that exceeds customer defined minimum specifications, especially benefits long-haul fleets that travel between diverse climates. Fuels with a lower than normal CP minimize the possibility of fuel gelling and undesirable downtime. Customers and/ or suppliers throughout the distribution network usually determine product cloud point values.

The trend of petroleum refiners optimizing fuel costs with online process control technology and thereby minimizing "product giveaway" may have a negative cold flow operability impact on fleets. Special care should be taken by long-haul fleets to ensure ULSD fuels meet their own specific cold flow operability needs throughout their regular routes. TMC advises fleets to employ a routine fuel testing program.

BIODIESEL

Biodiesel is a fuel composed of fatty acid methyl esters or fatty acid ethyl esters derived from renewable sources such as vegetable oil, animal fat and cooking oil. Esters are oxygenated organic compounds that can be used in compression ignition engines because many of their key properties are comparable to those of diesel fuel. "Soy Methyl Ester" diesel ("SME" or "SOME"), derived from soybean oil, is the most common biodiesel in the United States.

The most common process to manufacture biodiesel is esterification. The esterification process leaves residual alcohol that can lower the flash point of the biodiesel. The solvency may remove deposits from fuel tanks and lines causing filter plugging during initial use. TMC recommends that equipment users flush their fuel systems thoroughly with biodiesel prior to operational use and, replace fuel filters frequently in the early stages of operation.

The United States Department of Agriculture conducted studies of the "Low-Temperature Filterability Properties of Alternative Diesel Fuels from Vegetable Oils"[5] and demonstrated a nearly linear relationship between CP and low-temperature filterability of distillate/methyl ester blends. That is, cold-temperature operability limits predicted by dynamic filterability tests such as CFPP and LTFT were directly proportional to corresponding CP. This trend was observed for formulations containing 10 to 100 percent by volume methyl esters. More details regarding test methods can be viewed in the section **Petroleum Industry Cold Flow Testing Practices**.

Special care is required for neat biodiesel splash blending and storage issues in ambient conditions below 45°F. The National Renewable Energy Laboratory has performed extensive testing in this area. The typical cloud point of neat biodiesel ranges from 27°F to 54°F (-3°C to +12°C)[6].

Neat biodiesel and higher percentage biodiesel blends can cause a variety of engine performance problems especially filter plugging. A recent Engine Manufacturers Association (EMA) statement concluded that biodiesel blends up to a maximum of five percent (B5) should not cause engine or fuel system problems[7]. The engine manufacturer should be consulted if a higher percentage biodiesel blend is used.

Several standards organizations worldwide have recently adopted biodiesel specifications. Specifically, ASTM International introduced in 2001 a specification for biodiesel referenced as D6751[8]. In January 2005, the National Biodiesel Accreditation Program came into existence with the introduction of BQ9000. This biodiesel quality control system accredits qualified producers and certifies marketers who comply with specific quality control standards including those defined in ASTM D6751.

It is important that equipment users purchase biodiesel fuels that meet or exceed ASTM D6751 fuel quality test standards. As defined in ASTM D6751, one of the most critical tests for equipment users to be aware of is the fuel's CP value that is specified "by customer." It should be noted that there is no specification for a biodiesel/petroleum diesel blend. There is active work underway at ASTM to include a B5 biodiesel blend in D975. There is also a new ASTM standard in progress for a stand-alone B20 blend.

CONTAMINANTS THAT MAY IMPACT FUEL COLD FLOW

Water and Sediment

Diesel fuel should be clear in appearance and free of water and sediment. The presence of these materials generally indicates poor fuel handling practices. Water that freezes and forms ice crystals in fuel as well as other sediment can plug fuel filters; this may lead to fuel starvation and downtime. In addition, water can promote fuel system corrosion and microbial growth.

Microbial Growth

Microbial growth is the result of the presence of free water either in transport, storage or vehicle tanks.

Microbes do not live in fuel; they live in the interface that forms between the fuel and free water. The presence of microbes is indicative of contamination problems in the above areas. The presence of microbes can cause operational problems, corrosion, and sediment build-up in diesel fuel systems. Note however, the absence of microbes in fuel purchased at filling stations does not ensure the absence of microbes in fuel storage tanks or vehicle fuel systems.

Biodiesel fuel is an excellent medium for microbial growth. Inasmuch as water accelerates microbial growth and is naturally more prevalent in biodiesel fuels than in petroleum-based diesel fuels. For more detailed information on fuel handling, treatment and housekeeping guidelines refer to TMC RP 345, *Diesel Fuel Housekeeping Guidelines*[9].

FUEL COLD FLOW OPERABILITY RECOMMENDATIONS

When blending fuels, it's useful to know that No. 1 diesel is a refinery stock that is similar to kerosene and jet fuel. No. 1 diesel generally has excellent winter operability because the CP is extremely low, typically in the range of -40°F (-40°C) or even lower. However, there are several drawbacks with the use of No. 1 diesel. It has lower energy content, which affects fuel economy and power. In addition, the lubricity is generally poorer than No. 2 diesel, and the price is typically higher.

When considering modifying fuels, it makes good economical sense to optimize the use of No. 1 diesel and chemical additives. In order to achieve that result, it's important to test and modify fuels for specific ambient operating needs. Recently, portable cloud point analyzers for field applications have become available to assist fleets with fuel blending and/or knowing when cold flow improving additives may be required.

⚠ DANGER: Under no circumstances should gasoline be used for blending, this may present an explosion hazard.

As critical as it is to ensure that a fuel's cloud point is lower than the ambient temperature to avoid vehicle downtime, it's also important that No. 1 diesel not be over-used. To minimize the use of No. 1 diesel, one has to know the operability of the fuel before and after blending. A regular fuel analysis program plays an important role in the optimization process. In many situations, one may find that the operability of a seasonably adjusted winter fuel is suitable for a particular route without blending any No. 1 diesel at all. In the instance of using chemical additives to improve cold-flow, it's also important to ascertain the fuels operability so that one knows when or when not to use additives.

CP is an extremely important test as it detects the first onset of wax crystal formation. As illustrated in **Figures 1-9** there is a significant risk for untreated fuels of filter plugging and vehicle downtime when ambient temperature drops below a fuel's cloud point. Once plugged, the most effective way to remove wax crystals from a fuel system is to add heat.

The joint EMA/TMC recommendation[10] is a CP 4°C (7.2°F) below the minimum anticipated ambient temperature which the fuel is expected to operate. Consult with engine manufacturers technical bulletins for similar cold flow operability recommendations.

TMC recommends that equipment users adopt a fuel quality control testing program. Depending on its economic viability, testing may be performed in-house or with the cooperation of your fuel supplier. The net result of optimizing fuel blends or additive consumption will reduce a fleet's overall fuel cost.

To understand fully how fuels may be tested and modified for your own specific needs, it's important to understand the test methods employed within the petroleum refining and distribution industry.

PETROLEUM INDUSTRY COLD FLOW TESTING PRACTICES

There are four main cold flow property test methods in use, in alphabetical order they are:

1. **Cloud Point (ASTM D2500[11] manual method and D5771[12], D5772[13], D5773 automatic methods)—**CP is the temperature at which the smallest observable cluster of wax first appears upon cooling a fuel sample under prescribed conditions. It is these wax crystals that plug fuel filters. The CP of fuel will not be significantly lowered by chemical means. CP is a quality control test that has been in use by petroleum refiners for more than 50 years. In North America, a significant majority of petroleum refiners rely on the CP test for determining diesel fuel operability in cold weather. CP is the preferred test due to its fail-safe nature in predicting operability, quick analysis time and excellent precision. Besides CP, other operability test methods are available; how-

ever, their precision is significantly poorer and their test speed is slower, often making them impractical for field use.

2. **Cold Filter Plugging Point (ASTM D 6371[14])—**CFPP is the highest temperature at which a fixed volume of fuel fails to pass through a standardized filtration device in a specified amount of time when cooled under the conditions prescribed in this test method. CFPP is a dynamic test that utilizes plain weave stainless steel wire mesh gauze with a nominal aperture size of 45 µm (45 micron). The CFPP test was designed in 1960s for European light-duty vehicles. CFPP is primarily used to measure additized fuel operability improvements in diesel engines in winter months. The CFPP of a fuel is a good relative estimation of the lowest temperature at which a chemically treated fuel will give trouble-free flow in certain fuel systems. CFPP is typically lower than CP and higher than pour point. CFPP values that are more than 10°C colder than CP should be considered as suspect. The Worldwide Fuel Charter[15] recommends that CFPP values shall not exceed 10°C (18°F) below CP; therefore, CP is required for proper reference. Whether a fixed screen size of 45 microns can represent the filter pore range of today's diesel fuel systems remains debatable.

3. **Low Temperature Flow Test (ASTM D4539[16])**—LTFT involves a dynamic test procedure in which the temperature of a series of test specimens of fuel is lowered at a fixed cooling rate of 1°C/hour. Commencing at a desired test temperature and at each 2°F (1°C) interval thereafter, a separate specimen from the series is filtered through a 17µm screen until a minimum LTFT pass temperature is obtained. The minimum LTFT pass temperature is the lowest temperature, expressed as a multiple of 2°F (1°C), at which a fixed volume of the test specimen can be filtered in 60 seconds or less. The test method is especially useful for the evaluation of fuels containing flow-improving additives.

   Introduced in a 1981 Coordinated Research Council (CRC) study[17], LTFT results are the most conservative indicators of the low temperature flow performance of fuel in North American heavy-duty vehicles manufactured in that period of time. However, the length of the test makes it impractical for field use. Whether a fixed screen size of 17 microns can represent the filter pore range of today's diesel fuel systems vehicles remains debatable. For untreated fuels, LTFT agrees very well with CP.

4. **Pour Point (ASTM D97[18])—**Pour point (PP) is the lowest temperature expressed as a multiple of 5°F (3.0°C) at which the oil is observed to flow when cooled and examined under prescribed conditions. Diesel fuel at or below PP is beyond operability in any modern diesel engine. PP temperatures are below cloud point, cold filter plugging point and low temperature flow test. From a truck fleet perspective, PP applications involve the pumpability of fuels for distribution purposes. The PP of a

TABLE 1:
TEST TIME AND ACCURACY IN PREDICTING OPERABILITY

Test Type	CP	CFPP	LTFT	PP
Test time	3-30 minutes (a)	1-6 hours	24-48 hours	1-3 hours
Test reliability rate	98% (b)	64% (b)	89% (b)	13%(c)
Primary applications	Field & Laboratory	Laboratory	Laboratory	Laboratory
a. ASTM automatic methods D5771, D5772 and D5773. ASTM D2500 manual cloud point measurements range from one to three hours.				
b. Results obtained from the Coordinated Research Council Study CRC-528. A more recent study, SAE Technical Paper 2000-01-288319, produced very similar results.				
c. Results obtained from the Coordinated Research Council Study CRC-528.				

TABLE 2:
REPRODUCIBILITY OF ASTM
LOW TEMPERATURE PROPERTY TESTS

Test type	ASTM Crosscheck Testing Method Number	Reproducibility Between Laboratories °Celsius	Reproducibility (e) TMC conversion to °Fahrenheit
CP	D2500 (a)	4.3	7.7
	D5771 (b)	2.9	5.2
	D5772 (b)	2.9	5.2
	D5773 (b)	2.5	4.5
CFPP	D6371	3.0 to 4.2	5.4 to 7.6
LTFT	D4539 (c)	4.0	7.2
PP	D97 (d)	5.0 to 6.1	9.0 to 11.0
a. D2500 is a manual test method			
b. D5771, D5772 and D5773 are automatic cloud point test methods			
c. LTFT data, Coordinated Research Council study CRC-528. No recent LTFT data is available due to the methods limited use.			
d. Ongoing ASTM D97 crosscheck data.			
e. ASTM International temperature measurements are only expressed in degrees Celsius; for the benefit of the TMC membership this table represents a temperature conversion to degrees Fahrenheit.			

fuel may be modified by diluting with No. 1 (kerosene) diesel or by chemicals additives.

Table 1 offers an application comparison for the merits of each of these four test methods.

In terms of CP precision, ASTM has maintained an interlaboratory crosscheck program with more than 200 participating laboratories worldwide. **Table 2** illustrates the average precision expressed as reproducibility from 1997 to early 2002 for 16 diesel samples. In 2001, ASTM also began crosscheck tests on two un-additized samples for CFPP. A smaller reproducibility value is more desirable as it means the method is more precise.

REFERENCES

[1] ASTM Designation D5773, "Standard Test Method for Cloud Point of Petroleum Products (Constant Cooling Rate Method)." ASTM International 2005, http://www.astm.org

[2] ASTM Designation D975, "Standard Specification for Diesel Fuel Oil." ASTM International 2005, http://www.astm.org.

[3] TMC RP 304B.

[4] TMC RP 311.

[5] Dunn et. al. (1995; 1996) http://www.biodiesel.org/resources/reportsdatabase/reports/gen/19960801_gen-150.pdf

[6] National Renewable Energy Laboratory document NREL/TP-580-30004.

[7] http://www.enginemanufacturers.org/admin/library/upload/297.pdf

[8] ASTM Designation D6751 "Standard Specification for Biodiesel Fuel Blend Stock (B100) for Middle Distillate Fuels." ASTM International 2005, http://www.astm.org.

[9] TMC RP 345.

[10] "EMA Consensus Position: Joint EMA/TMC Pump Grade Specification for Premium Diesel Fuel," available from http://

www.enginemanufacturers.org/admin/library/upload/61.pdf

[11] ASTM Designation D2500 "Standard Test Method for Cloud Point of Petroleum Products." ASTM International 2005, http://www.astm.org.

[12] ASTM Designation D5771 "Standard Test Method for Cloud Point of Petroleum Products (Optical Detection Stepped Cooling Method)." ASTM International 2005, http://www.astm.org.

[13] ASTM Designation 5772 "Standard Test Method for Cloud Point of Petroleum Products (Linear Cooling Rate Method)." ASTM International 2005, http://www.astm.org.

[14] ASTM Designation D6371 "Standard Test Method for Cold Filter Plugging Point of Diesel and Heating Fuels." ASTM International 2005, http://www.astm.org

[15] "Worldwide Fuel Charter," December 2002. http://www.oica.net/htdocs/Main.htm

[16] ASTM Designation D4539 "Standard Test Method for Filterability of Diesel Fuels by Low-Temperature Flow Test (LTFT)." ASTM International 2005, http://www.astm.org

[17] "1981 CRC Diesel Fuel Low-Temperature Operability Field Test," Coordinated Research Council document CRC-528.

[18] ASTM Designation D97 "Standard Test Method for Pour Point of Petroleum Products." ASTM International 2005, http://www.astm.org

[19] J. Chandler and J. Zechman, "Low-Temperature Operability Limits of Late Model Heavy-Duty Diesel Trucks and the Effect Operability Additives and Changes to the Fuel Delivery System Have on Low-Temperature Performance," SAE Technical Paper 2000-01-2883.

Proposed RP 434v2 (T) **VMRS 002**

CAB AND SLEEPER INSULATION EFFICIENCY GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to establish guidelines for original equipment manufacturers (OEMs) for cab and sleeper insulation packages that maximize the performance of the heater ventilation and air conditioning systems to the satisfaction of the vehicle operator.

These guidelines include two levels of insulation packages. A "standard" package being an OEMs offering in all cabs and sleepers unless otherwise specified and a "premium" package offered as an option. If an OEM chooses to offer only one insulation package, it is recommended it meet the performance criterion of the premium offering.

This RP applies to Class 7-8 tractors with or without sleeper cabs.

TEST METHODOLOGY

The thermal rating test shall be conducted by using heating devices inside the vehicle compartment and by measuring the electrical input power required to maintain a constant interior-to-exterior temperature difference for a specified time. All testing shall be performed in accordance to TMC RP 422 with the results reported in R-value format. The tests will include interior trim, insulation, and cab/sleeper exterior materials as a complete package.

INSULATION MATERIAL

The insulation materials used may include, but are not limited to:

- Acoustical Urethane Foam,
- Faced Thermal-Acoustical Urethane Foam,
- Bun Stock Urethane Foam, Melamine foam,
- Polystyrene Rigid Foam (Styrofoam),
- Non-woven Polyester Fibers, and;
- Fiberglass.

In all cases the materials must comply with Federal Motor Vehicle Safety Standard (FMVSS) 302 flammability standards. Furthermore, TMC recommends Thermal or Solar glass be offered to customers for additional thermal transfer and ultraviolet ray protection.

INSULATION COVERAGE

TMC recommends that the insulation used in the cab, sleeper, and floor areas has a coverage factor satisfactory to meet the overall performance criteria established in this document.

STANDARD PACKAGE

TMC recommends that the standard cab and sleeper insulation package has an aggregate "R" value of greater than 4.2 and less than or equal to 4.6.

PREMIUM PACKAGE

It is recommended the premium cab and sleeper insulation package has an aggregate "R" value of greater than 4.6. This can include the use of thermal and or solar glass. Also, it is strongly recommended the insulation material be either single or double side faced to enhance thermal reflection.

Issued x/xxxx

Proposed RP 533 (T) **VMRS 053-006**

EMPLOYEE TOOL EXPENSE PROGRAM SELECTION GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

This Recommended Practice (RP) offers guidelines for evaluating and selecting tool expense programs for maintenance operations that require their employees/technicians to use their personal tools on the job.

INTRODUCTION

Technicians play an important role in building a successful trucking operation. What's more, technicians are becoming hard to recruit and retain in today's job marketplace. In fact, the U.S. Department of Labor forecasts an industry shortage of more than 38,000 commercial vehicle technicians through at least 2012.

One of the greatest challenges facing both prospective and professional technicians today is the cost of maintaining and acquiring tools. It is accepted practice in the commercial vehicle industry for technicians to supply their own tools on the job. However, tool expense programs can lessen this burden for their employees, as these programs can increase a technician's take home pay, through reimbursement of the cost of tools and tool use payments. This, in turn, benefits employers by being better able to recruit and retain technicians.

Determining the suitability of tool expense programs can be challenging. As a result, TMC has developed this RP to help fleet managers evaluate these programs for their operations.

HOW TOOL EXPENSE PROGRAMS WORK

Tool expense programs are suited for companies that require their technicians to supply and use their own personal tools to perform their job. Through a required "arms length transaction," typically performed by a third party administrator (TPA), the administrator evaluates the separation of the value of the employee's tools (tool pay) from the value of their labor (labor pay).

This separation assigns up to 35 percent of the original wage to tool pay. Therefore, the original pay to the technician is now paid in these two separate categories. These payments are made under separate agreement —which is provided by the TPA—between the employer and employee.

As a result, the employee receives labor pay separate from their tool pay. The tool expense portion of the employee's pay is a reimbursement for cost of, or a payment for the use of, the tools. The tool pay is fully and/or partially exempt from payroll tax, resulting in an increase in the technician's take home pay, and the company realizes an improved bottom line.

TPAs typically offer the following free standing tool expense programs:

- *Reimbursement for the acquisition cost of the technician's existing Inventory.* In this program, the tool pay portion is tax exempt. This exemption continues until the tool pay payments have depleted the acquisition cost of the technician's existing tool inventory. This value is provided by the technician.
- *Reimbursement for the acquisition cost of new tools after the technician has enrolled in the tool program.* In this program, the tool pay portion is tax exempt. This exemption continues until tool pay payments have depleted the acquisition cost of the technician's new tool.
- *Tool use expense.* In this program, the tool pay portion is exempt of employment taxes. This exemption is paid every pay period. Tool use expense payments may only be a portion of the tool pay, if there is a portion of the tool pay assigned to a cost reimbursement balance.

TOOL EXPENSE PROGRAM ADVANTAGES

Tool expense programs offer advantages for both employer and employee. Employers benefit through:

- Reduced taxes, which result in an increase in the company's bottom line.
- Increased take home pay for the employee.
- Increased employee retention.
- Increased opportunity to attract quality employees.
- Increased potential for participation in a 401(k)/retirement plan.

Employees benefit because of:

- Reduced taxes, which increase take home pay.
- Increased productivity, as savings are invested in more tools.
- Increased potential for participation in a 401(k)/retirement plan.
- a simplified tax return and additional savings with respect to the two percent gross wage deductible.

Tool expense programs can contribute significantly to employee retention. Employee retention reduces technician recruitment costs, training costs, and lost productivity. These programs can also be valuable in attracting new employees, too.

TOOL EXPENSE PROGRAM GUIDELINES AND RECOMMENDATIONS

When implementing a tool expense program, managers should consider the following recommendations and guidelines:

A. Tax Compliance Recommendations

TPAs rely on various tax codes and revenue rulings/interpretations to substantiate the tax compliance of their programs. In order to assess that the proper codes are used, fleet or service-dealer managers should:

- Ask for TPA's substantial authority opinion.
- Determine if the TPA's tax position is supported by a reputable accounting firm and tax law firm.
- Determine if the TPA or its legal/accounting representatives will defend any tax inquiry of the program at no cost its clients or technicians.

B. Payroll and IT Requirements

Determine if the TPA can provide a "one check" solution, or only a separate "second check" option. One-check solutions:

- Deliver tax savings to the employee's paycheck every pay period.
- Offer the technician the simplicity of receiving one check.
- Avoid the need for the accounting department to reconcile two checks.
- Eliminate the need to manage the delivery of two checks.
- Eliminate the risk of the second check not arriving on payday.
- Ensure and maintain the IRS-required "arms length transaction," since the TPA provides the tool pay calculation, and processes/delivers the tool pay data to the client.

In order for a TPA to determine the separation of tools from labor, and then process the tool pay, tool expense program clients must provide the following information:

- An employee information file, which allows the TPA to determine the pretax deduction, appropriate taxes, etc. This ensures that withholdings mirror payroll elections.
- Labor rates and hours for each technician, or the technician's gross performance pre-calculated each pay period will be exported from the payroll system and sent to the TPA.

Once it has calculated the separation and tool pay, the TPA will send back the gross tool pay , which will be either imported or manually entered into the payroll system. The payroll is then ready to be completed, or sent to the payroll vendor for completion. The tool and labor pay will be itemized on the employee's pay stub. In most cases, these imports and exports are created template reports, which require minimal effort to populate and exchange. These files are usually exchanged in comma-separated value (CSV) or Microsoft Excel spreadsheet format, which are created to accommodate the in-house payroll system or payroll vendor.

The TPA will electronically supply its clients with a detailed voucher every pay period for each technician. The voucher itemizes the difference between gross and net tool pay, which would consist of any applicable taxes and fees. Also contained in this report is tool cost reimbursement, balance remaining, year-to-date numbers, etc.

Other electronic pay period reports that a TPA can provide include an invoice, invoice detail, and check register. The invoice will reflect the fees and appropriate taxes, which will be electronically transferred from the company upon completion of the tool pay processing. The TPA submits any appropriate taxes on behalf of the technician.

NOTE: It is important to determine from the TPA when they expect to have the account funded for this electronic withdrawal, since some TPA's require this payment days in advance of the actual processing.

C. Benefit Packages

When implementing a tool expense program, a company should consider how the program will affect a technician's other employee benefits, such as a 401(k) program, etc. Each company will be unique in this regard and must review its specific benefits program.

For example, employee benefits which are a product of gross wage—such as 401(k) contributions, etc.—are typically driven off the benefit compensation clause which is defined in a company's benefits package. If the benefits are calculated on gross before deductions, then there may no effect on the benefits calculated. In the event the benefits are a product of W-2 wages, then the benefits will be affected, unless an adjustment to the employee's contribution is made to offset the W-2 reduction.

A company's matching 401(k) contribution may be affected by a tool program, resulting in reduced contributions in some instances. Again, the increase in contribution by the employee's percentage can adjust the base to readjust this contribution, while maintaining the substantial increase in take home pay to the technician.

Through an analysis provided by the TPA, this effect can be mitigated by an increase in the technician's percentage contribution, in order to achieve the same dollar contribution on the lower W-2 wage. An original six percent contribution by the technician would be increased to nine percent, in order to achieve the same dollar contribution.

NOTE: Consult your company's benefits administrator to provide guidance regarding tool expense programs and the qualifying compensation clause.

D. Social Security

Tool expense programs reduce FICA contributions because they are exempt of employment tax for both the company and its technicians. As a result, a technician could experience a reduction in Social Security retirement benefits. However, since benefits are calculated based on a 35-year average, there may be little or no reduction at all.

Considering that Social Security Administration estimates predict insolvency of the Social Security program by 2014, and a reduction to 73 percent of promised benefits, technicians would be wise to self-direct tax savings gained into their 401(k), and/or other retirement and savings opportunities. A TPA should provide, and review with technicians, a package which acknowledges and illustrates the possible reduction compared to the tremendous increase in retirement income derived from reinvesting some portion of the tool pay tax savings into an alternative investment, such as a 401(k).

E. Technician Enrollment Procedures

TPA's should work with company management to coordinate the program enrollment campaign. The TPA should conduct onsite enrollment meetings to ensure that the technicians understand the program, and encourage a high rate of enrollment. The TPA should provide all enrollment materials, including the enrollment form, which is a separate agreement between the company and the employee.

The TPA should provide a pre-enrollment kit for the technician to prepare their tool inventory. The TPA should present the program benefits at the enrollment meeting, answer all questions concerning the program, and enroll all interested technicians. The TPA should provide company management with a report acknowledging the enrollment results and detailing follow up efforts.

F. Program Maintenance

Program maintenance should primarily concentrate on new hires, tool purchase submissions, customer Service, etc. While each company will customize program plans with their TPA, most plans include:

- New hire packages—New Hires are typically enrolled as part of the employee new hire package, by whomever reviews these benefits with new technicians. For assistance regarding the program, new hires would call the TPA customer service center, which will explain program benefits and enrollment procedures. In some cases, a company will provide new hire reports to the TPA for follow up by the TPA's customer service department.
- New tool receipt submissions—New tool receipt submissions are always a challenge for technicians to maintain an organized system. The TPA should provide new tool packages

which offer technicians a simplified method to accumulate, organize and submit the receipts monthly, if not sooner, so the technician and the employer can realize the new tool acquisition benefit. Typically this is done with TPA-provided, self-addressed new tool envelopes, which technicians use to accumulate and submit their new tool receipts. The TPA should also maintain a calendar and send reminders to its client, to encourage timely submission of these envelopes.

- Customer service—Customer Service should be provided to answer any employer or employee questions.

G. Administrative Fees

Administrative fees are typically calculated as a percentage of tool pay. Administrative fees apply to the employer and the employee for administration of each program. Lower fees are assessed to the employer during the cost reimbursement phases, while higher fees are charged to the technicians. During the use program, the higher fee is charged to the employer and the lower fee to the technician.

Making a lower fee available to the technician during the use program is important in that it allows for a measurable savings to the technician. Without realizing some savings, technicians will not remain in the program. If the technician doesn't participate, there is no savings for the employer.

Proposed RP 653 (T) VMRS 036

FIFTH WHEEL COUPLING AND UNCOUPLING GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to provide technicians and drivers safe coupling and uncoupling procedures for fifth wheels of various manufacturer and application. This RP applies to fifth wheel devices used in typical Class 7-8 combination vehicle applications.

FIFTH WHEEL COUPLING PROCEDURE

1. Make sure that the fifth wheel is unlocked and is ready for coupling. (See **Figure 1**.)

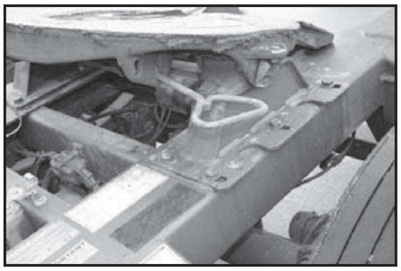


Figure 1

2. Chock the trailer wheels. Slowly back the tractor or dolly to the front of the trailer. Verify that there is the proper coupling height between the fifth wheel and the trailer upper-coupler. The recommended contact area should be 4-6" behind the pivot of the fifth wheel. Also verify that the kingpin is aligned with the locking channel. (See **Figure 2**.)

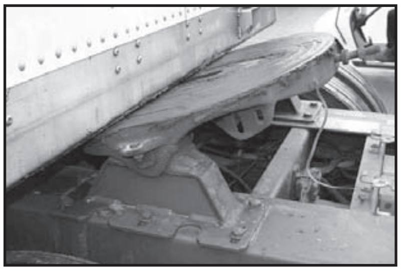
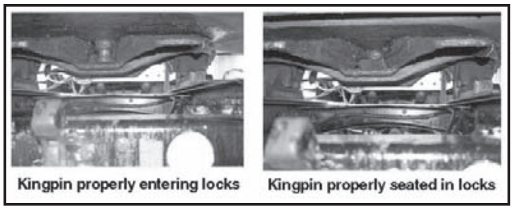


Figure 2

3. Slowly back the tractor or dolly into the trailer at idle speed, up to one mile per hour. Continue to couple the locks around the kingpin. (See **Figure 3**.)



Kingpin properly entering locks
Kingpin properly seated in locks

Figure 3

4. Properly attach the service and emergency air lines to the trailer gladhands. (See **Figure 4**.)

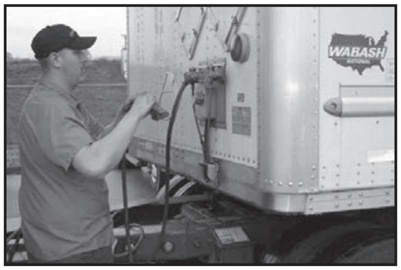


Figure 4

5. Connect the electrical supply cord to the trailer. (See **Figure 5**.)

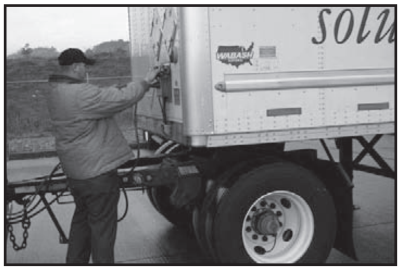


Figure 5

6. Retract the landing gear to full retraction. (See **Figure 6**.)

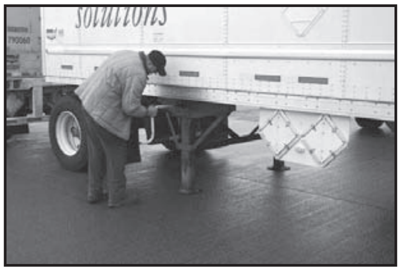


Figure 6

7. Properly stow dolly crank handle in holder. (See **Figure 7**.)

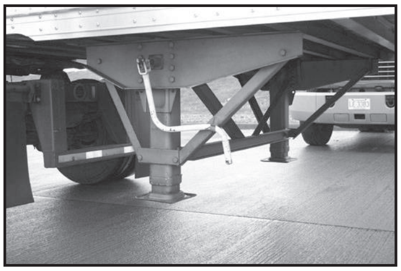


Figure 7

8. Before performing a pull test, check the following indications of a locked condition, based on the manufacturer of the fifth wheel. (See **Figures 8-12**.)

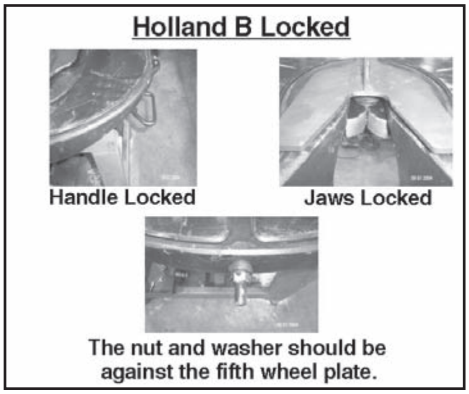


Figure 8

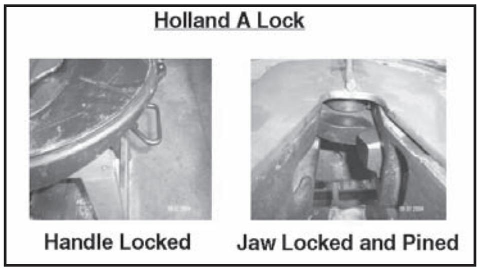


Figure 9

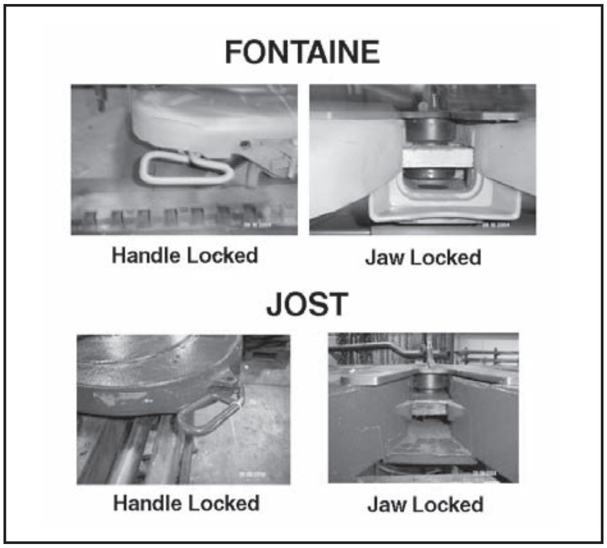


Figure 10

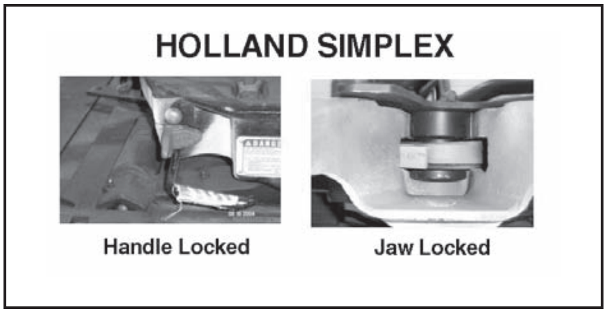


Figure 11

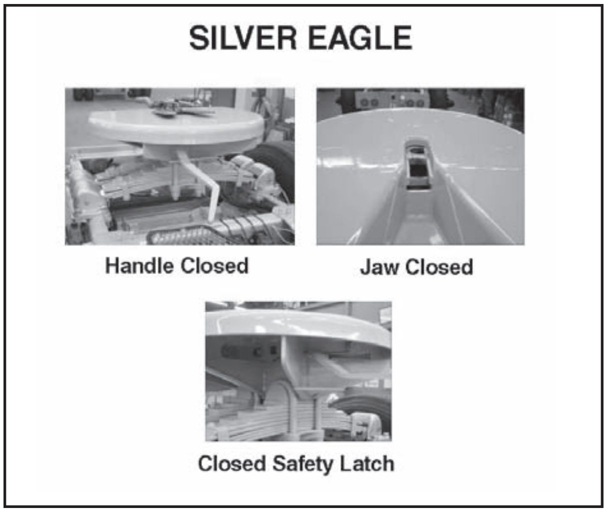


Figure 12

9. Now that you have verified that the fifth wheel is locked, check to see that there are zero gaps between the fifth wheel plate and the upper coupler on the trailer. (See **Figure 13**.)

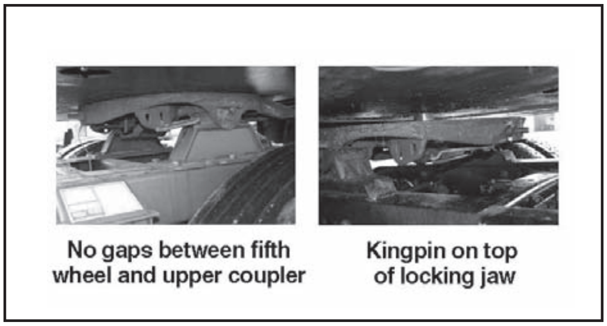

No gaps between fifth wheel and upper coupler
Kingpin on top of locking jaw

Figure 13

10. (Kingpin in front of fifth wheel) After checking that there are no gaps between the fifth wheel and the upper coupler, check to make sure the kingpin is not in front of the fifth wheel instead of engaged in the locking mechanism. (See **Figure 14**.)

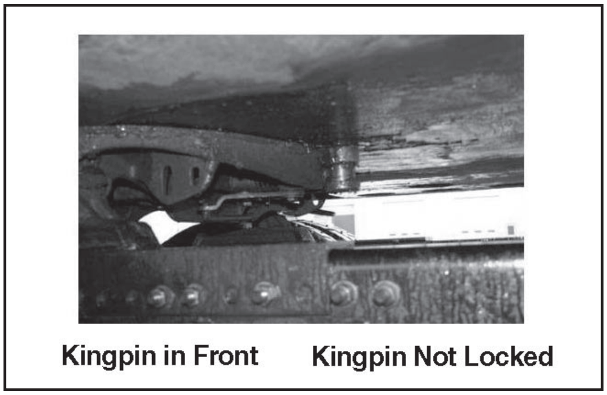


Figure 14

11. (Pull test) Remove the chocks from the trailer. Release the tractor brakes, leaving the trailer emergency brakes applied. Put tractor in gear and lightly pull the tractor forward to verify that the fifth wheel is properly locked.

FIFTH WHEEL UNCOUPLING PROCEDURE

1. Chock the trailer wheels and apply the trailer parking brakes. Lower the landing gear until a slight load is present in the crank handle. (See **Figure 15**.)

Figure 15

2. Properly stow dolly crank handle in holder. (See **Figure 16**.)

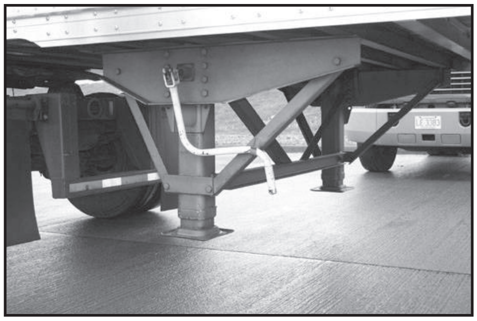


Figure 16

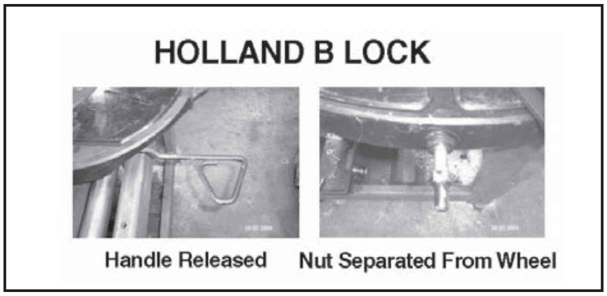
3. Disconnect the electrical plug from the trailer. Secure the electrical cord to the towing unit. (See **Figure 17**.)

Figure 17

4. Disconnect the service and emergency gladhads from the trailer. Secure the air lines to the towing unit. (See **Figure 18**.)

Figure 18

5. Before attempting to separate the tractor from the trailer, check for indications of an unlocked condition based on the manufacture of the fifth wheel. (See **Figures 19-21.) TMC** also recommends dumping the air on an air suspension system if so equipped.
6. Some manufacturers have secondary manual locks. The secondary lock must be released



HOLLAND B LOCK

Figure 19

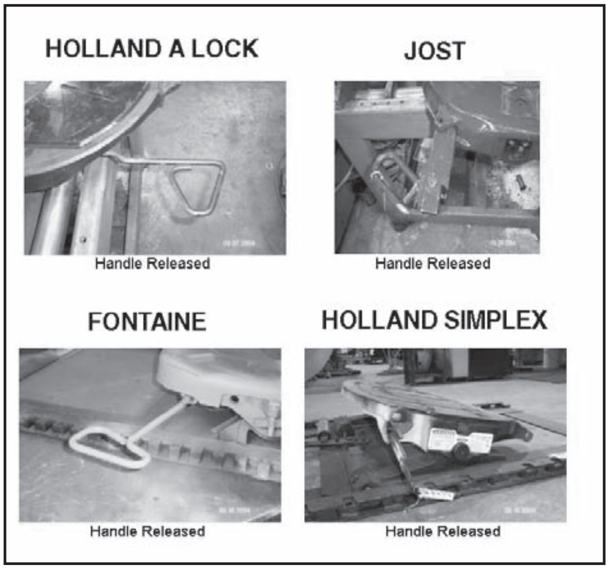
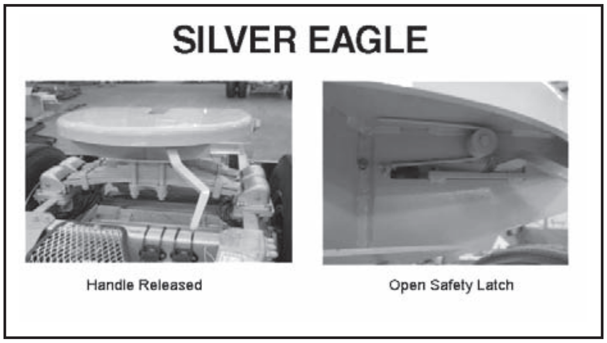


Figure 20



SILVER EAGLE

Figure 21

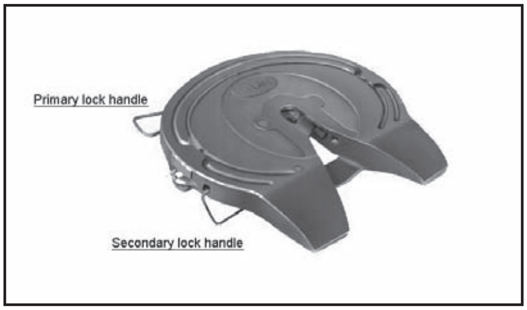
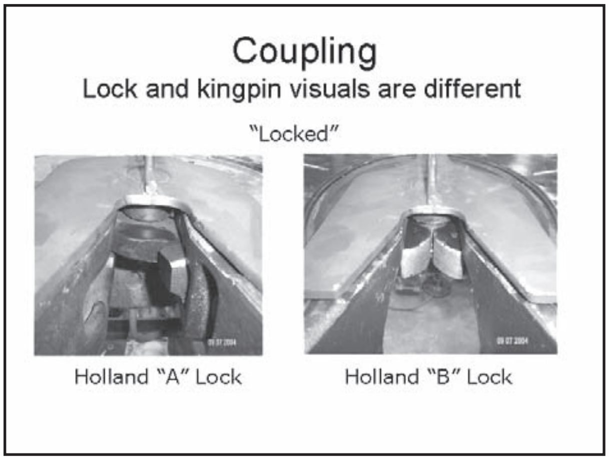


Figure 22

first before the main lock can be released or locked. (See **Figure 22**.)

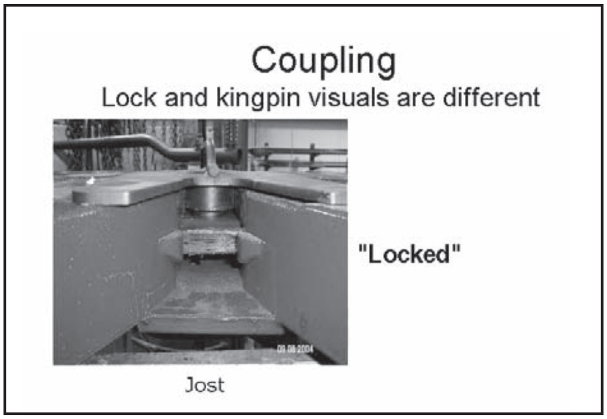
7. Lock and kingpin visuals differ in appearance from manufacturer to manufacturer and model to model. See **Figures 23-24** for specific examples. See **Figures 25-33** for examples of handles in locked and unlocked positions, varying by manufacturer.



Coupling

Lock and kingpin visuals are different

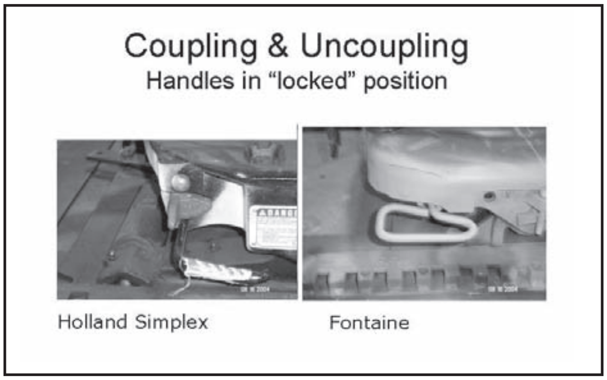
Figure 23



Coupling

Lock and kingpin visuals are different

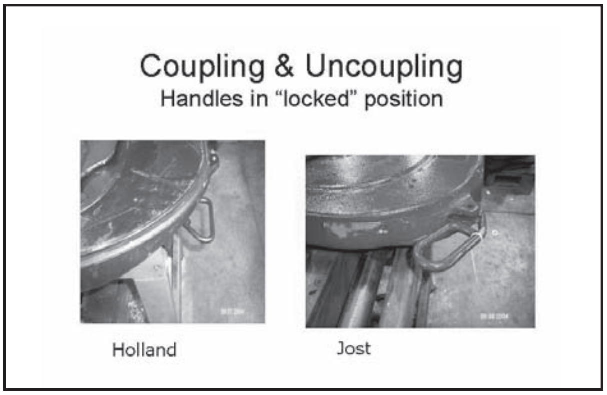
Figure 24



Coupling & Uncoupling

Handles in "locked" position

Figure 25

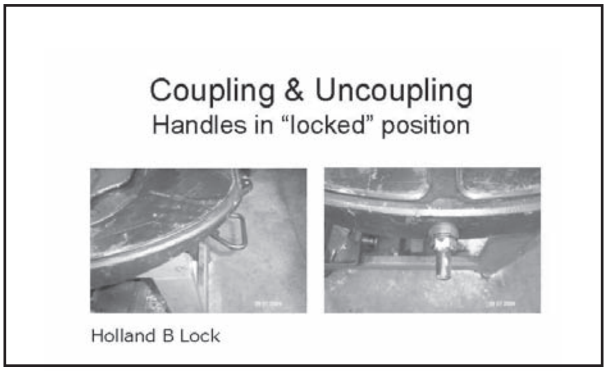


Coupling & Uncoupling
Handles in “locked” position

Holland

Jost

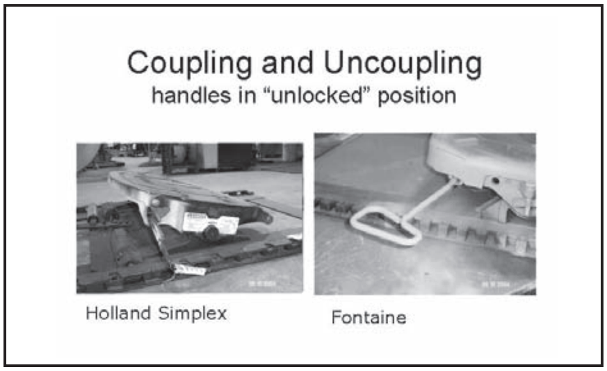
Figure 26



Coupling & Uncoupling
Handles in “locked” position

Holland B Lock

Figure 27

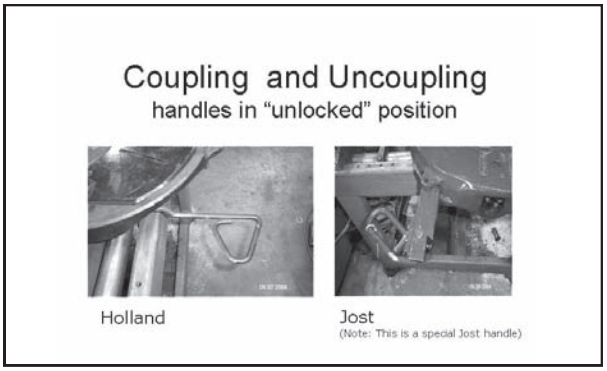


Coupling and Uncoupling
handles in “unlocked” position

Holland Simplex

Fontaine

Figure 28

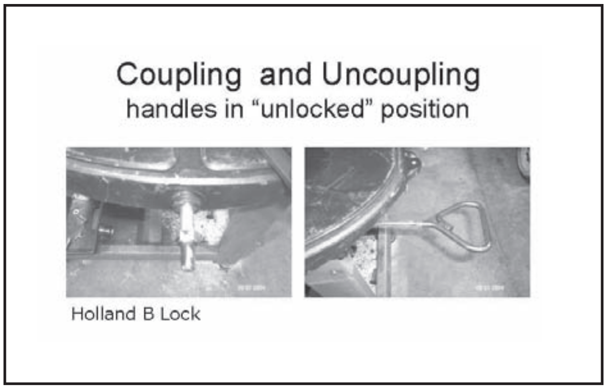


Coupling and Uncoupling
handles in “unlocked” position

Holland

Jost
(Note: This is a special Jost handle)

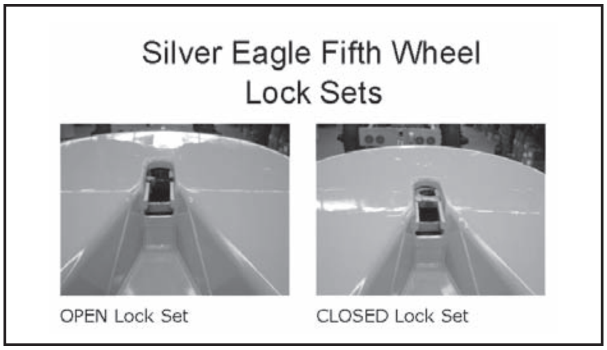
Figure 29



Coupling and Uncoupling
handles in “unlocked” position

Holland B Lock

Figure 30

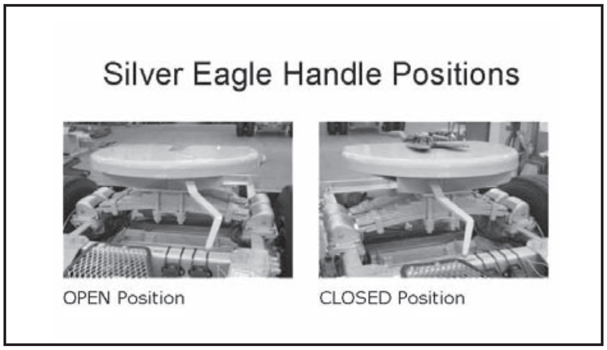


Silver Eagle Fifth Wheel
Lock Sets

OPEN Lock Set

CLOSED Lock Set

Figure 31

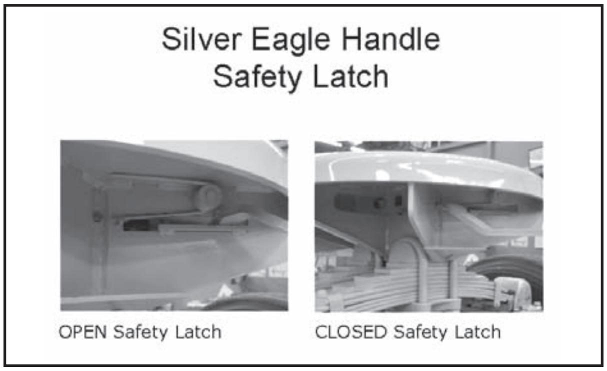


Silver Eagle Handle Positions

OPEN Position

CLOSED Position

Figure 32



Silver Eagle Handle
Safety Latch

OPEN Safety Latch

CLOSED Safety Latch

Figure 33

TMC Recommended Practice

Proposed RP 747 (T) **VMRS 059-004-002**

DRAWBAR EYE MAINTENANCE GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice is to provide maintenance, inspection and replacement guidelines for drawbar eyes. This RP applies to converter dollies used in heavy-duty combination vehicle applications.

DRAWBAR EYE MAINTENANCE GUIDELINES

Drawbar eyes, regardless of manufacturer, all have the same basic maintenance requirements and procedures. TMC recommends that drawbar eye maintenance and inspection be performed annually.

The following items should be part of a drawbar eye inspection:

- Check for worn, damaged or missing parts.
- Be sure the drawbar eye size is compatible with the coupling device on the tow vehicle, so that it fits securely into the coupling device.
- Inspect the coupling device on the tow vehicle for proper operation. Do not use any coupling device that does not operate properly. Take care not to damage coupling components during inspection, particularly during coupling and uncoupling procedures.
- Before operating, inspect for wear or damage and secure mounting. If wear exceeds 1/8" (0.125") from the original surface profile (see **Figure 1**), or if mounting is weakened, replace the drawbar eye and repair the mounting structure. If worn beyond 1/8", the wear is beyond the case hardening of the eye and rapid wear will result. Always measure for wear on the bottom of the eye (see **Figure 2**). For listing of drawbar eye diameters (when new) by part number, see **Table 1**. As an alternative, technicians may use a "go/no go gauge" to determine whether the drawbar eye should be replaced. These devices, as shown in **Figure 3**, are available from drawbar eye manufacturers.

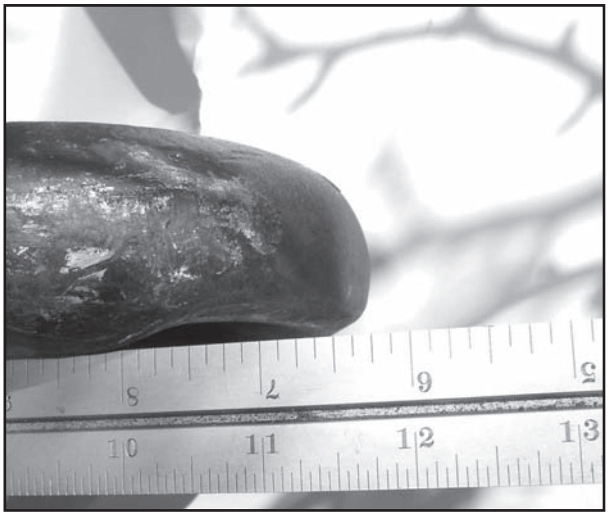


Figure 2

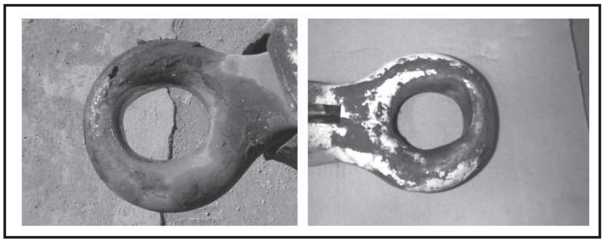


Figure 1

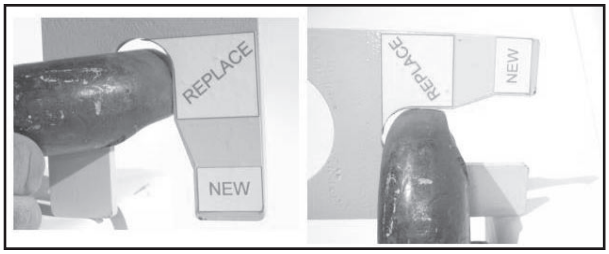


Figure 3

**TABLE 1:
DRAWBAR EYE DIAMETERS (WHEN NEW) BY PART NUMBER**

MODEL NO.	EYE SIZE (Inside Dia.)	SECTION	MODEL NO.	EYE SIZE (Inside Dia.)	SECTION
2	2"	7/8"	207	2 - 3/8"	1 - 11/16"
3	2 - 1/6"	1 - 1/2"	300	3"	1 - 5/8 x 1 - 3/4"
4	2 - 3/8"	1 - 5/8"	304	3"	1 - 5/8"
5	2 - 5/8"	1 - 3/4"	305	3"	1 - 5/8"
6	2 - 3/8"	1 - 11/16"	306	3"	1 - 3/4"
6A	2 - 3/8"	1 - 11/16"	307	3"	1 - 11/16"
8	2 - 3/8"	1 - 5/8"	309	3"	1 - 11/16"
11	2 - 3/8"	1 - 3/8"	405	1 - 3/8" min. 2 - 1/4" max.	1 - 11/16"
20	2 - 3/8"	1 - 5/8"	407	2 - 3/8"	1 - 5/8"
21	2 - 3/8"	1 - 11/16" x 1 - 5/8"	410	1 - 1/8" min. 3" max.	1 - 11/16"
22	2 - 3/8"	1 - 11/16" x 1 - 5/8"	DB-030DQ1	2.5"	1.25" x 1.5"
23	2 - 3/8"	1 - 11/16" x 1 - 5/8"	DB-040DQ1	2.5"	1.25" x 1.5"
107	2 - 3/8"	1 - 11/16" x 1 - 5/8"	DB-060FQ1	3"	1.63"
108	2 - 3/8"	1 - 11/16" x 1 - 5/8"	DB-1224	2.38"	1.63"
110	2 - 3/8"	1 - 11/16"	DB-1228-1	2.5"	1.25" x 1.5"
123	2 - 3/8"	1 - 5/8"	DB-1235-1	3"	1.63"
125JL	2 - 3/8"	1 - 5/8"	DB-1238	3"	1.63"
126JL	2 - 3/8"	1 - 5/8"	DB-1245-1	2.38"	1.63"
127F & 127	2 - 3/8"	1 - 11/16" x 1 - 5/8"	DB-1249-2H	3"	1.62"
200	2 - 3/8"	1 - 5/8"	DB-1249-49	3"	1.63"
203	2 - 1/2"	1 - 11/16" x 1 - 5/8"	DB-1250-3	2.5"	1.25" x 1.5"
205	2 - 3/8"	1 - 11/16"	DB-610-30	3"	1.63"

Proposed RP 748 (T) **VMRS Various**

TRAILER REFRIGERATION UNIT PREVENTIVE MAINTENANCE INSPECTION GUIDELINES

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Maintenance Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to offer guidelines for the preventive maintenance inspection of trailer refrigeration units used in heavy-duty commercial service. It has been developed from successful preventive maintenance programs currently used by several major manufacturers and motor carriers operating in North America with highly regarded reputations for excellence in vehicle maintenance.

INTRODUCTION

Preventive maintenance is the systematic and periodic inspection and servicing of vehicles and their components. The chief purpose of performing preventive maintenance is ensuring the operation of safe, roadworthy vehicles.

Through timely, quality preventive maintenance inspection (PMI), fleet operators should be able to detect, correct and prevent the development of costly vehicle systems breakdowns. In addition, a quality PMI program should provide fleets with the maximum return on their assets by attaining maximum component life of major vehicle systems before failure or replacement. A PMI program should also provide management with an instrument for predicting vehicle operational costs more accurately.

It's important for fleets to recognize that the PMI constitutes only a portion of a total preventive maintenance program. The total preventive maintenance program begins with a properly engineered and specified vehicle for the application at hand. Vehicle condition reports and accurate record keeping practices are also an integral part of managing and scheduling appropriate PMI's and service intervals. Finally, maintenance professionals must realize that quality preventive maintenance requires total commitment by all segments of company management. This document focuses on PMI procedures for trailer refrigeration units.

GENERAL COMMENTS

After determining the proper preventive maintenance inspection to be performed, shop management should assign PMI responsibilities to a trained and qualified technician. In order to enhance the technician's ability to identify small or premature defects, the unit should be cleaned before the inspection is started. By completing the inspection, performing the items listed on the PMI Record Sheet contained within this manual, and noting any defect or problems that are in need of repair, shop management can evaluate unit condition in a short period of time and schedule follow-up inspections and repairs intelligently, based on the availability of the vehicle, parts and labor.

A NOTE REGARDING PMI INTERVALS

This RP offers a specific PMI interval (annually or 3,000 engine hours) based on upon trailer refrigeration manufacturer recommendations. These intervals should be reviewed by fleet management based on vehicle specification and application. The maintenance history and component life of the vehicle are also factors to consider. Many of the programs that have been implemented by various fleets actually contain several different levels of PMI that are performed at a predetermined time or hour interval. Fleet management should review the inspection to determine which, if any, items might be eliminated or performed on a less frequent basis.

PMI FOLLOW UP

The depth of any PMI program can only be measured by program scheduling and repair of the defects found during the actual inspection. The PMI Record Sheet, included in this document, contains a section for noting defects or needed repairs, referenced by the line item from the PMI. The technician performing the PMI should initial each follow up item as the adjustments are completed. It's important to remember that completing the PMI itself is not sufficient; the quality of the overall PM program depends on completing the necessary follow-up repairs. Based on a sampling of various fleets that use a PMI program similar to the one described in this RP, TMC believes the average time to perform this procedure ranges between 60-90 minutes, depending on inspection frequency, content and vehicle specification.

TRAILER REFRIGERATION PMI INSTRUCTIONS

The following items should be completed during the PMI process. See **Table 1** for sample format of PM inspection sheet.

1. Pre trip unit. Cycle unit and record any fault codes.
2. Check for coolant, oil, fuel, or refrigerant fluid leaks. Then steam clean unit and recheck for leaks.
3. Clean debris from return screen, condenser, and evaporator coils.
4. Change engine oil and filter(s). Engine should be at operating temperature before draining oil. Take oil sample from used oil while draining. Use OEM recommended replacement oil.
5. Change fuel filter(s). Fill new filter(s) with clean diesel and lubricate filter o-ring prior to installing.
6. Check fuel lift pump intake screen for debris and replace sealing washers.
7. Check air filter restriction gauge (if equipped). Change filter if necessary and reset restriction gauge. If not equipped with a restriction gauge replace air filter annually or as required.
8. Check air intake system for loose clamps or holes in clean air piping.
9. Check water pump for excessive seepage. Loosen belt and check water pump bearing for looseness or rough bearing feel.
10. Check belt condition. Under normal driving conditions, rib cracking is generally the first condition to appear and is the early stages of a more severe failure mode called rib chunking. A common guideline suggests at least 80 percent of belt life has been depleted when a rib exhibits more than one crack per inch. Belt replacement is recommended at this point. Any portion missing from a rib in the undercord is an indication the belt is near the end of its service life and should be replaced. For more information, see TMC RP 320B, *Inspection, Maintenance and Tension of Accessory Belt Drive Systems*.) Check idler bearing for looseness or rough bearing feel. Adjust belt tension to manufacturer's spec with a tension gauge.
11. Check all hoses for leaks. Check hose clamps for condition and tightness.
12. Check radiator for corrosion, bugs, proper mounting, coolant level, and freeze protection to -34°F. Add 50/50 mixture only or adjust as necessary. Check pH level with test it. Treat as necessary.
13. Change ethylene glycol-based (conventional) coolant every two years; extended service life coolant every six years if not contaminated with conventional coolant/antifreeze. Pressure test cooling system and radiator cap.
14. Check battery cable connections. Clean and tighten the block ground. Check battery hold down. Load test battery.
15. Check all wiring for proper routing. Secure wiring harness away from moving parts or sharp objects.
16. Apply dielectric grease to all open connectors.
17. Check muffler and exhaust. Ensure clamps are tight and muffler is secure. Note any leaks, rust, etc.
18. Check that alternator mounting bracket bolts are tight. Check for cracks in brackets.
19. Check alternator voltage—minimum is 14.2 volts DC. If equipped, check alternator isolator for proper operation.
20. Check alternator brush length every 3,000 hours. Replace if less than 40 percent of new length. Inspect slip ring for excessive wear. If there is no access to brushes, remove alternator and replace brushes at 3,000 hours.
21. Check solenoid linkage and boot. Clean plunger and bore. Lube with white grease every 3,000 hours.

22 Check engine mounts for looseness or deteriorated rubber.

23. Check fan shaft and idler(s) for looseness, noise or lube leakage.
24. Check reefer panels, doors hinges, latch and trailer door switch.
25. Check compressor drive mechanism for security and wear. The engine should rotate less than 1/4" back and forth without turning the compressor. Any more than 1/4" means excessive wear.
26. Check refrigeration unit mounting bolt torque. Torque to manufacturer's specification.
27. Check damper door bushings for looseness.
28. Check solenoid and gearbox for proper operation.

29. Check manual defrost below 35°F box temperature. Damper door should close tightly or evaporator fan should stop.
30. Run defrost mode until completed. Defrost cycle should terminate in less than 15 minutes.
31. Check compressor oil level at compressor sight glass after finishing defrost cycle. Sight glass should be no more than 1/2 full.
32. Check defrost drains (kazoos) for presence, blockage and deterioration.
33. Check engine high and low RPM (manufacturer's spec).
34 Check air switch setting/operation with magnahelic gauge. Refer to manufacturer's specs.
35 Drain water from fuel tank and fuel/water separator.
36. Check fuel tank mounting brackets and straps for secure mounting.
37. Check fuel tank vent for blockage by removing the vent and blowing through it.
38. Check safety switches (oil, high discharge, coolant) for proper operation. Pull wire off switch to see if alarms work.
39. Check bulkhead condition.
37. Check air chutes for proper attachment, tears or deterioration.
38. Connect gauges and check high and low pressures for proper range. Refer to manufacturer's specs.
39. Download data recorder if equipped and store as required. Check date, time and proper time zone.
40. Apply completed PM sticker on inside door panel.

TRAILER REFRIGERATION PM INSPECTION SHEET
SINGLE TEMPERATURE UNIT

Refrigeration Unit #:	Unit Serial #:	Date:
Trailer #:	EE #:	Technician ID:
Make or Model:	Switch Hours:	Engine Hours:
Campaigns Due:	I/S Date:	

NOTE: Technician must initial box and mark appropriate response.
STATUS: ✔ = OK **A** = Adjust **R** = Repair **N/A** = Not Applicable

ITEM	STATUS	INITIALS	INSPECTION ITEM
1			Pre trip unit. Cycle unit and record any fault codes.
2			Check for coolant, oil, fuel, or refrigerant fluid leaks. Steam clean unit and recheck for leaks.
3			Clean debris from return screen, condenser, and evaporator coils.
4			Change engine oil and filter(s). Take sample.
5			Change fuel filter(s).
6			Check fuel lift pump intake screen.
7			Check air filter restriction. Change air filter if necessary.
8			Check air intake system for loose clamps or holes in clean air piping.
9			Check water pump.
10			Check belt condition.
11			Check hoses and clamps for condition and tightness.
12			Check radiator for corrosion, bugs, proper mounting, coolant level, and freeze protection (to -34°F). Check pH level.
13			Change coolant if needed. Pressure test cooling system and radiator cap.
14			Check battery cable connections. Clean and tighten the block ground. Check battery hold down. Load test battery.
15			Check all wiring for proper routing and securement.
16			Apply dielectric grease to all open connectors.
17			Check muffler and exhaust. Ensure clamps are tight; muffler is secure. Note leaks, rust, etc.
18			Check that alternator mounting bracket bolts are tight. Check for cracks in brackets.
19			Check alternator voltage—minimum is 14.2 volts DC. If equipped, check alternator isolator.
20			Check alternator brush length every 3,000 hours.
21			Check solenoid linkage and boot. Clean plunger and bore. Lube with white grease every 3,000 hours.
22			Check engine mounts for looseness or deteriorated rubber.
23			Check fan shaft and idler(s) for looseness, noise or lube leakage.
24			Check reefer panels, doors hinges, latch and trailer door switch.
25			Check compressor drive mechanism for security and wear.
26			Check refrigeration unit mounting bolt torque.
27			Check damper door bushings for looseness.
28			Check solenoid and gearbox for proper operation.
29			Check manual defrost below 35°F box temperature.
30			Run defrost mode until completed. Defrost cycle should terminate in less than 15 minutes.
31			Check compressor oil level at compressor sight glass after finishing defrost cycle.
32			Check defrost drains (kazoos) for presence, blockage and deterioration.
33			Check engine high and low RPM (manufacturer's spec).
34			Check air switch setting/operation with magnahelic gauge.
35			Drain water from fuel tank and fuel/water separator.
36			Check fuel tank mounting brackets and straps for secure mounting.
37			Check air chutes for proper attachment, tears or deterioration.
38			Connect gauges and check high and low pressures for proper range
39			Download and store data from recorder if so equipped. Check date, time and time zone.
40			Apply completed PM sticker on inside door panel.

TMC Recommended Practice

Proposed RP 802C(d)(T) **VMRS Various**

PROPOSED REVISION TO VMRS 2000— CODE KEY 79: POSITION CODE

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to update TMC RP 802C, *TMC/ATA Vehicle Maintenance Reporting Standards (VMRS)* through the revision of Code Key 79: Position Codes. This RP applies to the VMRS coding convention and all equipment types supported by VMRS 2000, the latest version of this standard.

INTRODUCTION

VMRS is a dynamic standard, changing to meet the needs of a dynamic industry. TMC is proposing the revision of VMRS Code Key 79: Position Codes. position codes. Code Key 79 is used to identify the position of a particular component on a vehicle, in cases where position is not an inherent element of the part itself. If the change is adopted, Code Key 79 would be updated as described in this RP.

EXISTING CODE STRUCTURE

Code Key 79 currently defines position through three Code Key subsets or elements:

- Code Key 79a: Side Code
- Code Key 79b: Orientation Code
- Code Key 79c: Sequence Code

Taken together, these subsets define position in an six digit code:

Example: 02-01-03 where:
02=Side Code, rear
05=Orientation, top/upper
03=Sequence, third

Code Key 79a Side Code

Side codes define the side or plane (front to rear, left to right, top to bottom) relative to the driver seat orientation.

Code	Position
00	Not Applicable
01	Front
02	Rear
03	Left
04	Right
05	Top/Upper
06	Bottom/Lower

Code Key 79b Orientation Code: Top to Bottom

This code represents a component's general orientation upon a "side" (front to rear, left to right, top to bottom relative to the driver's seat orientation).

Code	Position
00	Not Applicable
05	Top/Upper
06	Bottom/Lower
07	Center
08	Inner
09	Outer

Code Key 79c Sequence Code

The sequence code is based on a count (front to rear, left to right, top to bottom). When a circular pattern is involved, the upper/front most occurrences are defined as the number one (1) position and the sequence is counted in a clockwise direction.

Code	Position
00	Not Applicable
01	First
02	Second
03	Third
04	Fourth
05	Fifth
06	Sixth
07	Seventh
08	Eighth
09	Ninth
10	Tenth
11	Eleventh
Etc.	Etc.

PROPOSED CODE STRUCTURE

Under the proposed revision, the updated Code Key 79 would define position through two Code Key subsets or elements:

- Code Key 79a: Side Orientation Code
- Code Key 79b: Sequence Code

Taken together, these subsets would define position in a four-digit code:

Example: 02-01 where
02=Left Front
01=Position 1

Code Key 79a Side-Orientation Code

This data element combines the Side and Orientation elements into one element. Codes define the side and orientation (first—front to rear, second—left to right, third—top to bottom) relative to the driver seat orientation. Code must be considered in conjunction with Code Key 33.

Code	Position
00	Default
01	Left
02	Left Front
03	Left Front Upper/Top
04	Left Front Upper/Top Inside
05	Left Front Upper/Top Center
06	Left Front Upper/Top Outside
07	Left Front Lower/Bottom
08	Left Front Lower/Bottom Inside
09	Left Front Lower/Bottom Center
10	Left Front Lower/Bottom Outside
11	Left Rear
12	Left Rear Lower/Bottom
13	Left Rear Lower/Bottom Inside
14	Left Rear Lower/Bottom Center
15	Left Rear Lower/Bottom Outside
16	Left Rear Upper/Top
17	Left Rear Upper/Top Inside
18	Left Rear Upper/Top Center
19	Left Rear Upper/Top Outside
20	Left Inside
21	Left Center
22	Left Outside
23	Right
24	Right Front
25	Right Front Upper/Top
26	Right Front Upper/Top Inside
27	Right Front Upper/Top Center
28	Right Front Upper/Top Outside
29	Right Front Lower/Bottom
30	Right Front Lower/Bottom Inside
31	Right Front Lower/Bottom Center
32	Right Front Lower/Bottom Outside
33	Right Rear
34	Right Rear Lower/Bottom
35	Right Rear Lower/Bottom Inside
36	Right Rear Lower/Bottom Center
37	Right Rear Lower/Bottom Outside
38	Right Rear Upper/Top
39	Right Rear Upper/Top Inside
40	Right Rear Upper/Top Center
41	Right Rear Upper/Top Outside
42	Right Inside
43	Right Center
44	Right Outside
45	Upper/Top
46	Upper/Top Front
47	Upper/Top Front Inside
48	Upper/Top Front Center
49	Upper/Top Front Outside
50	Upper/Top Rear
51	Upper/Top Rear Inside
52	Upper/Top Rear Center
53	Upper/Top Rear Outside
54	Lower/Bottom
55	Lower/Bottom Front
56	Lower/Bottom Front Inside
57	Lower/Bottom Front Center
58	Lower/Bottom Front Outside
59	Lower/Bottom Rear
60	Lower/Bottom Rear Inside
61	Lower/Bottom Rear Center
62	Lower/Bottom Rear Outside
63	Center
64 - 79	reserved for future use
80	Axle Position Left
81	Axle Position Left Outer
82	Axle Position Left Outer & Inner
83	Axle Position Left Inner
84	Axle Position Right
85	Axle Position Right Outer
86	Axle Position Right Outer & Inner
87	Axle Position Right Inner
88	Axle Position All
89	Circular Position 12:00 Clockwise
90 - 99	reserved for future use

Code Key 79b Sequence Code

The sequence code is based on a count (front to rear, left to right, top to bottom). When a circular pattern is involved, the upper/front most occurrences are defined as the number one (1) position and the sequence is counted in a clockwise direction.

Code	Position
00	Position Not Applicable
01	Position 1
02	Position 2
03	Position 3
04	Position 4
05	Position 5
06	Position 6
07	Position 7
08	Position 8
09	Position 9
10	Position 10

11	Position 11
12	Position 12
13	Position 13
14	Position 14
15	Position 15
16	Position 16
17	Position 17
18	Position 18
19	Position 19
20	Position 20
21	Position 21
22	Position 22
23	Position 23
24	Position 24
25	Position 25
26	Position 26
27	Position 27
28	Position 28
29	Position 29
30	Position 30
31	Position 31
32	Position 32
33	Position 33
34	Position 34
35	Position 35
36	Position 36
37	Position 37
38	Position 38
39	Position 39
40	Position 40
41	Position 41
42	Position 42
43	Position 43
44	Position 44
45	Position 45
46	Position 46
47	Position 47
48	Position 48
49	Position 49
50	Position 50
51	Position 51
52	Position 52
53	Position 53
54	Position 54
55	Position 55
56	Position 56
57	Position 57
58	Position 58
59	Position 59
60	Position 60
61	Position 61
62	Position 62
63	Position 63
64	Position 64
65	Position 65
66	Position 66
67	Position 67
68	Position 68
69	Position 69
70	Position 70
71	Position 71
72	Position 72
73	Position 73
74	Position 74
75	Position 75
76	Position 76
77	Position 77
78	Position 78
79	Position 79
80	Position 80
81	Position 81
82	Position 82
83	Position 83
84	Position 84
85	Position 85
86	Position 86
87	Position 87
88	Position 88
89	Position 89
90	Position 90
91	Position 91
92	Position 92
93	Position 93
94	Position 94
95	Position 95
96	Position 96
97	Position 97
98	Position 98
99	Position 99

EXAMPLES

The following position codes serve as examples for identifying position for an rear axle shaft.

022-002-001 Shaft - Axle

Code	**Description**
80 - 01	Axle Position Left, Position 1 (first axle)
80 - 02	Axle Position Left, Position 2 (second axle)
80 - 03	Axle Position Left, Position 3 (third axle)
84 - 01	Axle Position Right, Position 1 (first axle)
84 - 02	Axle Position Right, Position 2 (second axle)
84 - 03	Axle Position Right, Position 3 (third axle)

The following codes serve as examples for identifying position for a front wheel bearing assembly.

018-001-002 Bearing Assembly - Front Wheel, Outer

Code	**Description**
81 - 01	Axle Position Left Outer, Position 1 (first axle)
81 - 02	Axle Position Left Outer, Position 2 (second axle)
85 - 01	Axle Position Right Outer, Position 1 (first axle)
85 - 02	Axle Position Right Outer, Position 2 (second axle)

Recommended Practice

Proposed RP 1210B (T)

VMRS 053

WINDOWS™ COMMUNICATION API

TABLE OF CONTENTS

PREFACE 5
1. INTRODUCTION 5
1.1 PURPOSE AND SCOPE 5
1.2 RP 1210 BACKGROUND 5
1.3 BACKWARDS COMPATIBILITY 5
1.4 ERROR CODE 142 "ERR HARDWARE NOT RESPONDING" AND BACKWARDS COMPATIBILITY 5
1.5 OPERATING SYSTEMS AND SUPPORT 6
1.6 RP 1210B COMPLIANCE DEFINED 6
1.7 DOCUMENT CONVENTIONS 6
1.8 TECHNICAL DISCLAIMER 6
2. HIGH-LEVEL RP 1210 INTERFACE CONCEPT 7
3. HIGH-LEVEL DESIGN 7
4. FUNCTIONAL SPECIFICATION 7
4.1 REQUIRED HIGH-LEVEL FUNCTIONALITY 7
4.2 API FILE NAMING CONVENTIONS 7
4.2.1 Legacy Notes Concerning File Names 8
4.3 MESSAGE SEND/RECEIVE NOTIFICATION (LOCKING/NON-BLOCKING) 8
4.4 NOTES ON J1939 AND ISO 15765 COEXISTENCE 8
4.6 NOTES ON CAN 8
5. RP 1210B OVERVIEW FOR THE APPLICATION DEVELOPER 9
5.1 API INTERFACE AT A GLANCE 9
5.2 STEP 1—APPLICATION PARSING OF THE RP1210 INI FILES 9
5.3 STEP 2—ASKING USER WHICH ADAPTER TO USE (OR AUTO-DETECTING) 9
5.4 STEP 3—ATTACHING TO THE DLL AND GETTING FUNCTION POINTERS 10
5.5 STEP 4— CONNECTING TO THE DATA BUS 11
5.6 STEP 5—ALLOW RECEPTION OF MESSAGES 11
5.7 STEP 6—READ A MESSAGE 12
5.8 STEP 7—SEND A MESSAGE 12
5.9 STEP 8—CLOSING THE INTERFACE AND FREEING WINDOWS RESOURCES 13
6. RP 1210B VARIABLE AND INITIAL STATES 13
6.1 TABLE OF VARIABLES 13
6.1.1 EchoMode 13
6.1.2 J1708Mode and RP 1210B Enhancements 14
6.1.3 FilterStates 14
6.1.4 Blocking Mode 14
6.1.5 Blocking Timeout 14
6.1.6 J1708/J1939/CAN/J1850/ISO 15765 Filter Clarifications 14
6.1.7 J1708 Parameter ID (PID) Filtering 14
6.1.8 Inclusive and Exclusive Filtering 14
7. RP 1210B REQUIRED FUNCTIONS 15
8. CLARIFICATIONS FOR APPLICATION DEVELOPERS 15
8.1 UNEXPECTED VEHICLE INTERFACE ADAPTER DISCONNECT FROM VEHICLE OR COMPUTER 15
8.2 FILTER STATE AMBIGUITIES 15
8.3 APPLICATION DEVELOPER SUMMARY 15
8.4 INI FILE VERSUS REGISTRY 16
8.5 RP 1210B COMPLIANCE TESTING 16
9. COMPLIANCE FOR API/VDA VENDORS 16
9.1 STANDARD FUNCTIONS 16
9.2 RP1210_SENDCOMMAND FUNCTIONS 17
10. RP 1210 FUNCTION CALLS 17
10.1 RP1210_CLIENTCONNECT PROTOTYPE 17
10.1.1 RP1210_ClientConnect Prototype 18
10.1.2 RP1210_ClientConnect Parameters 18

10.1.3 RP1210_ClientConnect Return Value 19
10.1.4 RP1210_ClientConnect Notes 19
10.1.4.1 Multiple CAN Protocols on One CAN Channel; Multiple Baud Rates (J1939/J1708) 20
10.1.5 Protocol Connection String 20
10.1.5.1 Using TCP/IP as a Transport Layer for a Protocol 20
10.1.5.2 Using a Modem as a Transport Layer for a Protocol 21
10.1.5.3 SETTING CAN BAUD RATE on RP1210_ClientConnect 21
10.1.5.4 SETTING J1708 BAUD RATE on RP1210_ClientConnect (fpchProtocol) 22
10.1.5.5 SETTING J1939 BAUD RATE on RP1210_ClientConnect (fpchProtocol) 22
10.1.5.6 SETTING ISO15765-2 BAUD RATE on RP1210_ClientConnect 22
10.2 RP1210_CLIENTDISCONNECT 22
10.2.1 RP1210_ClientDisconnect Prototype 23
10.2.2 RP1210_ClientDisconnect Parameters 23
10.2.3 RP1210_ClientDisconnect Return Value 23
10.2.4 RP1210_ClientDisconnect Notes 23
10.3 RP1210_SENDMESSAGE 23
10.3.1 RP1210_SendMessage Prototype 24
10.3.2 RP1210_SendMessage Parameters 24
10.3.3 RP1210_SendMessage Return Value 24
10.3.4 RP1210_SendMessage Message Definitions (fpchClientMessage) 22
10.3.5 Formatting a J1708 Message for RP1210_SendMessage 22
10.3.6 Formatting a J1939 Message for RP1210_SendMessage 23
10.3.7 Formatting a CAN Message for RP1210_SendMessage 23
10.3.8 Formatting a J1850 Message for RP1210_SendMessage 24
10.3.9 Formatting an ISO 15765 Message for RP1210_SendMessage 24
10.4 RP1210_READMESSAGE 25
10.4.1 RP1210_ReadMessage Prototype 25
10.4.2. RP1210_ReadMessage Parameters 26
10.4.3 RP1210_ReadMessage Return Value 26
10.5 RP1210_READMESSAGE MESSAGE DEFINITIONS 27
10.5.1 The J1708 (and PLC4TRUCKS) Message Returned From RP1210_ReadMessage 27
10.5.2 The J1939 Message Returned From RP1210_ReadMessage 27
10.5.3 The CAN Message Returned From RP1210_ReadMessage 28
10.5.4 The J1850 Message Returned From RP1210_ReadMessage 29
10.5.5 The ISO15765 Message Returned From RP1210_ReadMessage 29
10.6 RP1210_READVERSION 31
10.6.1 RP1210_ReadVersion Prototype 31
10.6.2 RP1210_ ReadVersion Parameters 31
10.6.3 RP1210_ ReadVersion Return Value 32
10.7 RP1210_READDETAILEDVERSION 32
10.7.1 RP1210_ReadDetailedVersion Prototype 32
10.7.2 RP1210_ ReadDetailedVersion Parameters 32
10.7.3 RP1210_ ReadDetailedVersion Return Value 33
10.8 RP1210_GETERRORMSG 33
10.8.1 RP1210_GetErrorMsg Prototype 33
10.8.2 RP1210_GetErrorMsg Parameters 33
10.8.3 RP1210_GetErrorMsg Return Value 33
10.9 RP1210_GETLASTERRORMSG 34
10.9.1 RP1210_GetLastErrorMsg Prototype 34
10.9.2 RP1210_GetLastErrorMsg Parameters 34
10.9.3 RP1210_GetLastErrorMsg Return Value 34
10.10 RP1210_GetHardwareStatus 35
10.10.1 RP1210_GetHardwareStatus Prototype 35
10.10.2 RP1210_GetHardwareStatus Parameters 35
10.10.3 Information Returned in fpchClientInfo 36
10.10.3.1 Hardware Status (Bytes 0-1) 36
10.10.3.2 J1939 (Bytes 2-3) 36
10.10.3.3 J1708 (Bytes 4-5) 36
10.10.3.4 CAN (Bytes 6-7) 36
10.10.3.5 J1850 (Bytes 8-9) 37
10.10.3.6 Vendor 1, 2, 3 (Bytes 10/11, 12/13, 14/15) 37
10.10.3.7 ISO 15765 (Bytes 16-17) 37
10.10.4 RP1210_GetHardwareStatus Return Value 37
10.11 RP1210_SENDCOMMAND 37
10.11.1 RP1210_SendCommand Prototype 38
10.11.2 RP1210_ SendCommand Parameters 38

10.11.3 Values for the nCommandNumber Parameter 39
10.11.4 RP1210_ Reset_Device 39
10.11.4.1 RP1210_ Reset_Device Return Value 40
10.11.5. RP1210_Set_All_Filters_States_To_Pass 40
10.11.5.1 RP1210_ Set_All_Filter_States_To_Pass Return Value 40
10.11.6 RP1210_Set_Message_Filtering_For_J1939 40
10.11.6.1 J1939 Filter Parameter Frame 41
10.11.6.2 RP1210_Set_Message_Filtering_For_J1939 Return Value 41
10.11.7 RP1210_Set_Message_Filtering_For_CAN 41
10.11.7.1 CAN Filter Parameter Frame 42
10.11.7.2 Standard CAN (11-bit Identifier) Example 42
10.11.7.3 RP1210_Set_Message_Filtering_For_CAN Return Value 42
10.11.8 RP1210_Set_Message_Filtering_For_J1708 43
10.11.8.1 RP1210_Set_Message_Filtering_For_J1708 Return Value 43
10.11.9 RP1210_Set_Message_Filtering_For_J1850 43
10.11.9.1 RP1210_Set_Message_Filtering_For_J1850 Return Value 44
10.11.10. RP1210_Set_Message_Filtering_For_ISO15765 44
10.11.10.1 ISO15765 Filter Parameter Frame 45
10.11.10.2 RP1210_Set_Message_Filtering_For_ISO15765 Return Value 45
10.11.11 RP1210_Generic_Driver_Command 45
10.11.11.1 RP1210_Generic_Driver_Command Return Value 46
10.11.12 RP1210_Set_J1708_Mode 46
10.11.12.1 RP1210_Set_J1708_Mode Return Value 47
10.11.13 RP1210_Echo_Transmitted_Messages 47
10.11.13.1 RP1210_Echo_Transmitted_Messages Return Value 47
10.11.14 RP1210_Set_All_Filters_States_to_Discard 47
10.11.14.1 RP1210_ Set_All_Filter_States_To_Discard Return Value 48
10.11.15 RP1210_Set_Message_Receive 48
10.11.15.1 RP1210_ Set_Message_Receive Return Value 48
10.11.16 RP1210_Protect_J1939_Address 48
10.11.16.1 J1939 Address Claim Frame 49
10.11.16.2 RP1210_ Protect_J1939_Address Return Value 49
10.11.17 RP1210_Release_J1939_Address 50
10.11.17.1 J1939 Address Claim Frame 50
10.11.17.2 RP1210_ Release_J1939_Address Return Value 50
10.11.18 RP1210_Set_Broadcast_For_J1708/CAN/J1939/J1850/ISO15765 51
10.11.18.1 Broadcast List Entry (fpchClientCommand) 51
10.11.18.2 MESSAGE_TIME_TUPLE Record 52
10.11.18.3 List Behaviors 52
10.11.18.4 Return Value, All Function Calls 53
10.11.19 RP1210_Set_BlockTimeout 53
10.11.19.1 RP1210_Set_BlockTimeout Return Value 53
10.11.20 RP1210_Set_J1708_Baud 53
10.11.20.1 RP1210_Set_J1708_Baud Return Value 54
10.11.21 RP1210_Set_J1939_Baud 54
10.11.21.1 RP1210_Set_J1939_Baud Return Value 54
10.11.22 RP1210_Set_J1708/J1939/J1850/ISO15765/CAN_Filter_Type 55
10.11.22.1 RP1210_Set_J1708/J1939/J1850/ISO15765/CAN_Filter_Type Return Value 55
10.11.22.2 Example (RP1210A = Inclusive) 55
10.11.22.3 Example (RP1210B = Exclusive) 56
10.11.23 RP1210_Set_J1939_Interpacket_Time 56
10.11.23.1 RP1210_Set_J1939_Interpacket_Time Return Value 56
10.11.24 RP1210_Set_MaxErrorMsgSize 56
10.11.24.1 RP1210_Set_MaxErrorMsgSize Return Value 57
10.11.25 RP1210_Disallow_Further_Connections 57
10.11.25.1 RP1210_Disallow_Further_Connections Return Value 57
10.11.26 RP1210_Set_ISO15765_Flow_Control 57
10.11.26.1 ISO15765 Flow Control Parameter Frame 58
10.11.26.2 RP1210_Set_ISO15765_Flow_Control Return Value 59
10.11.27 RP1210_Clear_ISO15765_Flow_Control 59
10.11.27.1 RP1210_Clear_ISO15765_Flow_Control Return Value 59
10.11.28 RP1210_Set_ISO15765_Link_Type 60
10.11.28.1 RP1210_Set_ISO15765_Link_Type 60
11. INI FILE FORMAT 61
11.1 FILE FORMAT AND PARSING OF THE RP1210 INI FILES 61
11.2 VENDOR API INSTALLATION PROGRAM INSTRUCTIONS AND NOTES 62
11.2.1 Using and/or Creating the RP121032.INI File 62

11.2.2 Using and/or Creating the Vendor INI File 62
11.2.3The [VendorInformation] Section of the INI File 63
11.2.4 The [DeviceInformationdx] Section of the INI File 65
11.2.5 The [ProtocolInformationpx] Section of the INI File 65
11.3 STANDARDIZATION OF THE VENDOR INI FILE 66
11.3.1 Standardized Structure of INI File for Devices with Multiple Ports 66
12. DEBUG FILE FORMAT 72
13. TERMINOLOGY 73

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

This document describes a standardized interface—TMC's RP 1210 Windows™ Communication Application Program Interface (API)—for personal computer (PC) to on-vehicle ECU communications under the Microsoft Windows™ family of operating systems. This Recommended Practice (RP) establishes a standard API interface between the physical datalink (J1708/J1587/J1922, PLC, J1850, ISO15765, CAN, J1939) vehicle datalink dapaters (VDAs) and Windows™-based software applications for the PC.

Current and future hardware requirements have been carefully considered in developing this software API. This promotes the development of software applications for fleet maintenance, ECU reprogramming, and vehicle diagnostics using a standard common software interface. Anyone is welcome to employ this RP in implementing software systems for ECU communications or for development of VDAs.

A list of terms and accompanying definitions used in this document appears in **Section 13.**

1.2 RP 1210 BACKGROUND

This RP has undergone two prior revisions, the original RP 1210 and RP 1210A. This revision, called RP1210B, presents mostly enhancements and additions to RP 1210A as requested by OEMs, software development firms specializing in onboard vehicle communications, as well as manufacturers of VDAs.

The most significant changes that this version includes is the removal of support for Windows 3.1 and the addition of the ISO 15765 protocol. Also, the J1850 protocol, which did not get defined completely, will be fully defined in this version (note that only three-byte headers will be supported). Vendors will no longer be required to support and/or test the features of the API that were Windows 3.1 specific.

RP 1210A has been widely adopted and there are many (over 125 and growing monthly) RP1210A-compliant applications and adapters on the market, as of April 2006.

1.3 BACKWARDS COMPATIBILITY

RP1210B remains backward compatible with RP1210A, with the exception of dropping Windows 3.1 support. However future versions of RP1210 may not be backwards compatible with this or any previous version.

1.4 ERROR CODE 142 "ERR HARDWARE NOT RESPONDING" AND BACKWARDS COMPATIBILITY

Between the release of RP1210A and RP1210B, VDA vendors have made remarkable progress in adapter technology (USB, wireless, TCP/IP, etc). Some adapters are now powered by the PC (as is the case with USB adapters), or they could be powered by other means such as an internal battery, alternating current (AC), etc. These types of "non-traditional" powered adapters have caused some concerns within TMC's RP 1210 Update Task Force.

The question/issue that was dealt with is as follows:

- On a non-vehicle-battery-powered device, how does an API handle the case where the "physical" connection disappears (the link to the truck) but the "virtual" connection persists (VDA to PC link)? Do we create another error code in addition to 142 (ERR HARDWARE NOT RESPONDING), or do we work something else into the specification?

Statements and discussions included:

- Many RP 1210A applications are "keyed in" to getting a 142 Error Code back stating that something outside of the PC has gone wrong. Adding another error code could possibly cause confusion for RP 1210A-compliant applications that are running on an RP 1210B-compliant API.
- There needs to be a way to tell the user, in more detail, what is going on in an RP 1210B-compliant application running on top of an RP 1210B-compliant API.

TMC has resolved this issue as follows:

- Error Code 142 will remain the primary error code associated with hardware problems. In this way, RP 1210A-compliant applications will continue to function as normal (no need for porting or rewrite).
- TMC is adding a new function call called "RP1210_*GetLastErrorMsg* (short, int *, char *). This command will allow RP 1210B-compliant applications to get extended error information from the RP 1210B-compliant APIs

about the specific reason a 142 (or any other code) was returned.

The following deals with RP 1210B adapters and applications:

- An application can determine whether it is physically connected to the data bus by repeatedly calling RP1210_*GetHardwareStatus* and checking the correct bits to determine if the link has been active within the past second (note the changes in RP 1210B to the RP1210_*GetHardwareStatus* function). For RP 1210B applications working with an RP 1210A adapter, the operation of this is vendor-specific because it was implemented differently among RP 1210A VDA vendors.
- RP1210B applications should no longer depend solely on receiving the 142 Error Code and should instead use the RP1210_*GetHardwareStatus* function.

See also the section on RP1210_*GetLastErrorMsg*.

1.5 OPERATING SYSTEMS AND SUPPORT

As of April 2006, the following Windows™ operating systems were in common use.

- Windows 95
- Windows 98
- Windows Millennium Edition (ME)
- Windows XP Home Edition (XP-Home)
- Windows NT
- Windows 2000
- Windows XP Professional Edition (XP-Pro)
- Windows Server 2003
- Windows Vista

Manufacturers of software applications and vehicle communication adapters “may,” but are not required to, support any specific combination of the above operating systems. Consult specific software application and datalink adapter manuals to determine which operating systems are supported.

Windows 3.1 and all functionalities associated specifically with it have been removed from this RP. However datalink adapter or software application vendors may still provide support for Windows 3.1.

1.6 RP 1210B COMPLIANCE DEFINED

In versions prior to RP 1210B, no mention was given to the term “RP1210 compliant.” This ultimately lead to some confusion in the marketplace as to what the term actually meant. This section defines what it means to be an RP 1210B-compliant VDA and what it means to have an RP 1210B-compliant software application.

In general, any RP 1210B-compliant application should work with any RP 1210B-compliant vehicle interface adapter for their commonly supported operating systems and protocols without needing “work around code” to deal with the eccentricities of a specific adapter. This allows:

1. Vehicle interface adapters to be totally transparent to the application programmer. No portion of the interface should be tied to any specific method or hardware configuration.
2. Software product differentiation on a proprietary basis while still maintaining consistency between hardware interface devices.

RP 1210B-compliance does not require that a software application or vehicle interface adapter vendor must, will, or should, support all operating systems and protocols.

An RP 1210B-compliant VDA is:

- Any communications adapter that implements and specifically follows the API presented herein for that vendor’s supported operating systems and protocols.

An RP 1210B-compliant software application is:

- Any software application that implements and specifically follows the API presented herein for that vendor’s supported operating systems and protocols.

See the section in this document entitled **Compliance for API Vendors** which outlines all functions and sub-parameters that must be implemented for RP 1210B compliance.

1.7 DOCUMENT CONVENTIONS

Anything related to actual software code (i.e., examples and code snippets) or INI files will be in the Courier font. The following is an example:

```
if ( ( hRP1210DLL = LoadLibrary( szDLLName ) ) == NULL )
 {
   printf("Error:  LoadLibrary( %s ) failed! \n", szDLLName );
   exit(1);
 }
```

1.8 TECHNICAL DISCLAIMER

The intention and sole purpose of RP 1210B is to provide only an API and in no way endorses, recommends, approves, promotes or favors the preferential treatment of any vendor over another.

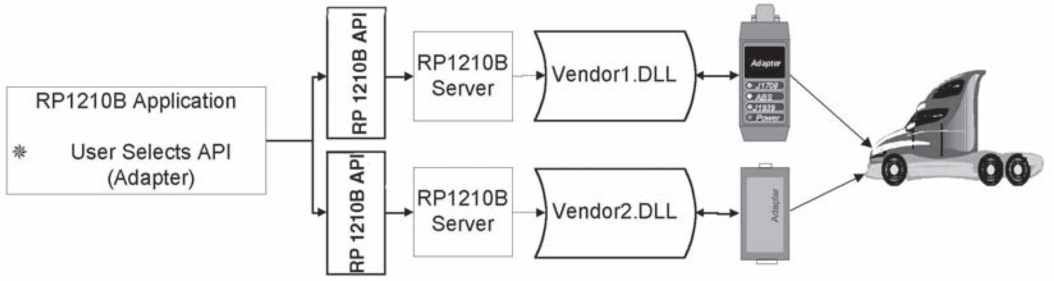


Fig. 4-1: TMC RP 1210B Interface Concept Implementation Model

2. HIGH-LEVEL RP 1210 INTERFACE CONCEPT

Figure 4-1 illustrates how the RP 1210B model is implemented.

- The RP1210B software developer is interested in only the "RP1210B Application" box.
- The RP1210B VDA manufacturer is responsible for the API, the Server, the resultant DLL (which is ultimately responsible for changing the vehicle protocol into API command responses) and the physical interface device .

3. HIGH-LEVEL DESIGN

This section provides a high-level design description of the API, which will be implemented as a Windows™ Dynamic Link Library or DLL (hereafter referred to as the API DLL or casually as the "API"). The main purpose of this DLL is to provide a generalized procedure-call interface between any hardware-specific drivers on the PC and the applications running on that PC.

Each vendor implementation must have built-in interface(s) with the underlying physical layers (hardware) supported by the particular vendor. Information about the vendor DLLs present as well as the protocols and devices supported by each implementation can be retrieved through INI files.

The end-user (or sometimes the application program) will choose the adapter to be used. Then the application will load that DLL through the use of the Windows™ LoadLibrary function, and will locate the specific functions (defined herein) through calls to the Windows™ *GetProcAddress* function.

4. FUNCTIONAL SPECIFICATION

4.1 REQUIRED HIGH-LEVEL FUNCTIONALITY

Multiple Client—The API must support a minimum of 16 clients. This is a clarification made in the upgrade from RP 1210A to RP 1210B.

Send/Receive Message Buffering—Since Microsoft Windows™ is event-driven, an API satisfying this RP must support this feature. There must be implicit support for synchronous as well as asynchronous communication.

Initialization/Reset—The application software must be able to initialize and reset the hardware through API-supplied functions at any time. If the hardware does not support mid-stream initialization or resetting, the API-supplied function shall provide a return value to this effect.

Time-Stamping of Messages—Since messages received from the vehicle datalink are buffered by the API, there must be a time-stamp associated with each message to resolve ambiguity and establish an order of precedence (for the application software) between successive messages. This function would ordinarily be implemented in the driver software interfacing with the API. The timestamp is four bytes in length and is in Big Endian or Motorola format. See the RP1210_*ReadMessage* function for the definition and layout of the timestamp. The resolution of the timestamp is implementation-dependent and is given in the INI file.

Version Reporting—The API shall be able to report its software version information. In RP 1210B, there is a separate, more detailed function that has been added.

Message Filtering—The API must support message filtering, typically performed by the lower-level interfaces, as specified in the definition of the RP1210_*SendCommand* function. An API implementation shall provide each client with exactly those messages as are specified in its filter specification.

4.2 API FILE NAMING CONVENTIONS

In general, each vendor will provide a different name implementation of the API and a number of these implementations could simultaneously reside on the same PC. No vendor shall name its implementation "RP1210.DLL" or "RP121032.DLL".

Implementations will follow the following format (using VENDRX for a vendor name):

File Type	Resultant File Name
INI File	VENDRX32.ini
Header (.H) File	VENDRX32.h
DLL File	VENDRX32.dll

Each VDA vendor's implementation will have all three of the aforementioned file types (ini, h, dll).

4.2.1 Legacy Notes Concerning File Names

Some older legacy implementations on Windows 3.1 (16-bit architecture) may still exist and may reside on the PC that an API is be installed on. The filenames would be VENDRX16.INI, VENDRX16.H, VENDRX16.DLL as well as a main INI file of RP1210.INI.

Although long file names are supported by all of the operating systems mentioned herein, an API DLL shall still be named in accordance with the file allocation table (FAT) file system naming convention (which allows up to eight characters for the file name and three characters for the extension with no spaces anywhere).

4.3 MESSAGE SEND/RECEIVE NOTIFICATION (LOCKING/NON-BLOCKING)

The aforementioned operating systems all support pre-emptive multitasking. This is accomplished through the use of threads (sometimes called "child processes"). A thread describes a path of execution within a process or application. It could be thought of as a basic unit of CPU utilization.

A process or an application can spawn other subprocesses as additional threads. Since each application has a primary thread automatically associated with it, the operating system divides CPU time between threads rather than applications. This provides for efficient CPU utilization and cleanly supported multitasking.

Multi-threading shall be supported through the mechanisms of a blocking read and a blocking send. There are flags in various commands that allows the user to specify whether the functions should execute in a blocked or a non-blocked manner.

4.4 NOTES ON J1939 AND ISO 15765 COEXISTENCE

RP 1210B supports ISO 15765-2 messages that appear either on a generic-CAN bus or on a J1939 datalink.

To handle the case where a connection is made to a J1939 datalink that could contain ISO 15765-2 traffic, the following strategies are used to filter ISO 15765-2 messages from a J1939 connection, and J1939 messages from an ISO 15765-2 client connection.

When a client sele-cts the "J1939" protocol, the RP 1210B API will automatically filter (remove) messages from the stream when:

- The J1939 Extended Data Page bit (EDP – formerly Reserved bit) and Data Page bit are both set, as defined in ISO 15765-3, or
- The message is on PGN 52480, 52736, 55808 or 56404, as defined in ISO 15765-2 Annex A.

When a client selects the "ISO 15765" protocol, the application is required to specify the type of datalink which in turn defines the mechanism that is used to filter J1939 messages from being seen by the ISO 15765 client connection. The datalink type is specified via the RP1210_*SendCommand* "RP1210_*Set_ISO15765_Link_Type*", and the options available are:

- ***Generic CAN bus***. No filtering of J1939 messaging occurs. This is the default.
- ***J1939 and ISO15765 coexistence as defined in J1939/21 and ISO15765-3***. Messages that do not have bit 25 and bit 24 of the CAN identifier both set will be filtered (removed) from the data stream.
- ***J1939 and ISO15765 coexistence as defined in ISO15765-2 Annex A***. Messages that are not on PGNs 52480, 52736, 55808 or 56404 will be filtered (removed) from the data stream.

The filtering introduced by the RP1210_*Set_ISO15765_Link_Type* is in addition to the filtering that can be user defined with the RP1210_*Set_Message_Filtering_For_ISO15765* command.

4.6 NOTES ON CAN

All CAN ID's are assumed to be in Big Endian format unless otherwise specified.

5.0 RP 1210B OVERVIEW FOR THE APPLICATION DEVELOPER

The following sections step the application developer through what is necessary to get their application up and going quickly.

5.1 API INTERFACE AT A GLANCE

TMC's RP 1210 API provides a defined INI file and a set of functions so that PC applications can perform send and receive operations through the DLL, to and from the vehicle's ECU's. In simplistic terms, the API provides an "open," "read," "write," and "close," interface. The following provides a simple look at what is required to send and read messages using the API interface:

Function Name	**Description**
Various	Parse the INI files for various information. For example, does the VDA support the needed protocol?
LoadLibrary (…)	Open the VDA API's DLL.
GetProcAddress(…)	Get pointers to the RP1210 functions within the VDA API's DLL.
RP1210_ClientConnect (…)	Get a "logical" connection to the vehicle data bus.
RP1210_SendCommand(…)	Allow messages to pass through the API.
RP1210_SendMessage (…)	Send a message.
RP1210_ReadMessage (…)	Read a message.
RP1210_ClientDisconnect (…)	Close the logical connection to the data bus.
FreeLibrary(…)	Close the VDA API's DLL.

5.2 STEP 1—APPLICATION PARSING OF THE RP1210 INI FILES

The first step that an application must perform is the parsing of the RP1210 INI files to find:

- Which adapters are present.
- Which of those adapters support the protocol and speed needed.

5.3 STEP 2—ASKING USER WHICH ADAPTER TO USE (OR AUTO-DETECTING)

The second step that an application must perform is either asking the user which adapter to use, or going through an auto-detect process to determine which adapter is present. Although auto-detection is a nice feature for an application to have, TMC recommends that the application initially bring up a VDA selection dialog box allowing the user to select the adapter. This information should include the following dialog box items:

- Vendor Name
- Adapter
- Protocol
- Protocol Connect Speed (this is an RP1210B new and advanced feature)
- TCP/IP Address and Port parameters if TCP/IP will be used as a transport layer for the protocol (this is an RP1210B new and advanced feature)
- Telephone number to dial if a modem will be used as a transport layer for the protocol (this is an RP1210B new and advanced feature)

Optionally, the VDA selection dialog box might have a check box to save the current adapter selection so that the user does not have to select the adapter a second time.

NOTE ON APPLICATIONS DOING AUTO-DETECTION (NOT RECOMMENDED) FOR VDA's

- Some application vendors have attempted, and failed, in algorithms to automatically detect the adapter that is physically attached to the computer. This has caused support nightmares for the API vendors, even if that API vendor did not write the application themselves. Some API's are just not capable of validating a specific device is connected to the PC. Also, in order to auto-detect, some API's require that the VDA be powered up and connected to the vehicle prior to performing an RP1210_*ClientConnect*.
- A new variable will be present in the INI file that will tell the application developer whether or not that the associated API devices are capable of being auto-detected. See the "**AutoDetectCapable**" variable in the INI file section of this document.

5.4 STEP 3—ATTACHING TO THE DLL AND GETTING FUNCTION POINTERS

The third step that an application must perform, after asking the user to choose the VDA, is to load the appropriate API DLL and resolve references to the DLL-supplied functions. This is accomplished by using the Window API functions, *LoadLibrary*, *GetProcAddress* and *FreeLibrary* (see the Windows API SDK reference for the details of these functions).

NOTE: If developing a 32-bit Visual Basic Application, the names defined should be as the Alias in the appropriate *Public Declare Function* statement.

Below is a simple example of loading the DLL and getting pointers to the RP1210-defined functions:

```
typedef short (WINAPI *fxRP1210_ClientConnect)          ( HWND, short, char *, long,
long, short );
typedef short (WINAPI *fxRP1210_ClientDisconnect)      ( short );
typedef short (WINAPI *fxRP1210_SendMessage)           ( short, char*, short, short, short );
typedef short (WINAPI *fxRP1210_ReadMessage)           ( short, char*, short, short );
typedef short (WINAPI *fxRP1210_SendCommand)           ( short, short, char*, short );
typedef void  (WINAPI *fxRP1210_ReadVersion)           ( char*, char*, char*, char* );
typedef short (WINAPI *fxRP1210_GetErrorMsg)           ( short, char*, short, short );
typedef short (WINAPI *fxRP1210_GetLastErrorMsg)       ( short, int *, char* );
typedef short (WINAPI *fxRP1210_GetHardwareStatus)     ( short, char* );
typedef short (WINAPI *fxRP1210_ReadDetailedVersion)   ( short, char*, char*, char* );

/*GLOBAL VARIABLE*/ fxRP1210_ClientConnect            pRP1210_ClientConnect           = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_ClientDisconnect         pRP1210_ClientDisconnect        = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_ReadMessage              pRP1210_ReadMessage             = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_SendMessage              pRP1210_SendMessage             = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_SendCommand              pRP1210_SendCommand             = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_ReadVersion              pRP1210_ReadVersion             = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_GetErrorMsg              pRP1210_GetErrorMsg             = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_GetLastErrorMsg          pRP1210_GetLastErrorMsg         = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_GetHardwareStatus        pRP1210_GetHardwareStatus       = NULL;
/*GLOBAL VARIABLE*/ fxRP1210_ReadDetailedVersion      pRP1210_ReadDetailedVersion     = NULL;
/*GLOBAL VARIABLE*/ HANDLE                            hRP1210DLL                      = NULL;

  if ( ( hRP1210DLL = LoadLibrary( szDLLName ) ) == NULL )
    {
      printf(“Error:  LoadLibrary( %s ) failed! \n”, szDLLName );
      exit(1);
    }

  pRP1210_ClientConnect = (fxRP1210_ClientConnect)(
                                                   GetProcAddress(
                                                                  hRP1210DLL,
                                                                  “RP1210_ClientConnect”
                                                                  )
                                                   );

  if ( pRP1210_ClientConnect == NULL )
    {
      pRP1210_ClientConnect = (fxRP1210_ClientConnect)(
                                                       GetProcAddress(
                                                                      hRP1210DLL,
                                                                      “_RP1210_ClientConnect@24”
                                                                      )
                                                       );
        if ( pRP1210_ClientConnect == NULL )
        {
          printf(“Error:  Could not find procedure \”%s\” in DLL!\n”,
                 “RP1210_ClientConnect” );
          exit(1);
        }
    }
```

5.5 STEP 4— CONNECTING TO THE DATA BUS

The last step that an application must perform, after loading all of the functions, is to connect through the VDA to the data bus and allow messages to be read. Below is a snippet of code:

```
nClientID = pRP1210_ClientConnect(
                                  NULL_WINDOW,
                                  nUserDevice,
                                  szProtocolInUse,
                                  lUserTXBuffer,
                                  lUserRXBuffer,
                                  0
                                  );



if ( ( nClientID >= 0 ) && ( nClientID <= 127 ) )
  {
    printf(“Connect OK. DLL [%s] Device [%d] Protocol [%s] ID = %d\n”,
           szDLLName,
           nUserDevice,
           szProtocolInUse,
           nClientID );

    nCurrentConnectState      = CONNECTED;


    return nClientID;
  }
else
  {
    nCurrentConnectState      = NOT_CONNECTED;
    PrintRP1210Error( “RP1210_ClientConnect”, nClientID );
    exit(1);
  }
```

5.6 STEP 5—ALLOW RECEPTION OF MESSAGES

After successfully connecting to the VDA, the next step is to enable reception of messages. When the call to *RP1210_ClientConnect* is made, the API does not allow messages to come through. This gives applications time to spin their read/write threads, etc. Below is a snippet of code that allows messages to pass:

```
nRetVal = pRP1210_SendCommand(
                              RP1210_Set_All_Filters_States_to_Pass,
                              nClientID,
                              (char *) NULL,
                              0
                              );
```

```
if( nRetVal != 0 )
  {
    PrintRP1210Error( “RP1210_Set_All_Filters_States_to_Pass”, nRetVal );
  }
else
  {
    printf(“RP1210_Set_All_Filters_States_to_Pass OK \n” );
  }
```

5.7 STEP 6—READ A MESSAGE

After successfully setting filters to allow messages, the application may now read messages. Below is a snippet of code that reads a message (with blocking):

```
nRetVal = pRP1210_ReadMessage( nClientID,
                              (char*) &ucTxRxBuffer[0],
                              sizeof(ucTxRxBuffer),
                              BLOCKING_IO );

if( nRetVal <= 0 )
  {
    PrintRP1210Error( “RP1210_ReadMessage”, (short) -nRetVal );
  }
else
  {
    // Process the message
  }
```

5.8 STEP 7—SEND A MESSAGE

After the *RP1210_ClientConnect*, the application may also send messages. Below is a snippet of code that sends a J1708 message (with blocking):

```
nRetVal = pRP1210_SendMessage(
                              nClientID,
                              (char *) &ucTxRxBuffer[0],
                              nMessageSize,
                              NULL_WINDOW,
                              BLOCKING_IO
                              );

if( nRetVal != 0 )
  {
    PrintRP1210Error( “RP1210_SendMessage”, nRetVal );
  }
else
  {
    printf(“Message sent successfully!\n”);
  }
```

5.9 STEP 8—CLOSING THE INTERFACE AND FREEING WINDOWS RESOURCES

When the program is finished and ready to exit, the application must close the interface and free the DLL. Below is a snippet of code that does that:

```
if ( pRP1210_ClientDisconnect( nClientID ) != 0 )
  {
    PrintRP1210Error( "RP1210_ClientDisconnect", (short) nRetVal );
  }

if ( FreeLibrary( (HMODULE) hRP1210DLL ) == 0 )
  {
    printf( "FreeLibrary() call fails.\n");
```

6.0 RP 1210B VARIABLE AND INITIAL STATES

6.1 TABLE OF VARIABLES

The following is a table of variables that describe the RP 1210 connection. Each API/VDA vendor will implement each variable with its initial state as defined. Different interpretations within RP 1210A VDA vendors have caused problems for application developers in the previous versions RP 1210 and RP1210A. Also, each application must call *RP1210_ClientConnect* before issuing any commands to the API via *RP1210_SendCommand*.

The following are the different RP 1210A variables that an application program must track. Implementation of these variables has been different among RP 1210A vendors and has prompted application developers to write "work-a-rounds" into their code:

Variable/Topic	Initial State on Connect	Description
EchoMode	Off	See EchoMode paragraph.
J1708Mode	Converted Mode	See J1708Mode paragraph.
FilterStates	All Filters to Discard	See FilterStates paragraph.
MessageReceive	On	See FilterStates paragraph.
BlockingMode	Blocking	See BlockingMode paragraph.
BlockingTimeout	Infinite	See BlockingTimeout paragraph.
J1939Packetize	On	API packetizes J1939 transport protocol messages for the application.
J1708FilterMode	Inclusive	New variable. See Inclusive/Exclusive Filtering paragraph.
J1939FilterMode	Inclusive	New variable. See Inclusive/Exclusive Filtering paragraph.
CANFilterMode	Inclusive	New variable. See Inclusive/Exclusive Filtering paragraph.
J1850FilterMode	Inclusive	New variable. See Inclusive/Exclusive Filtering paragraph.
ISO15765FilterMode	Inclusive	New variable. See Inclusive/Exclusive Filtering paragraph.

6.1.1 EchoMode

RP1210A vendors have implemented **EchoMode** differently (which affects the "Echo Byte" on received messages). Some vendors believe that "any" message sent should be treated as a "generic message appearing on the datalink" and, therefore, should be returned during a call to *RP1210_ReadMessage*. Other vendors return a sent message only when EchoMode is turned on and ensure that the message has an ECHO byte = 1.

Since the majority of vendors only return the sent message when EchoMode=1, any RP1210B compliant API will only return a message sent by "*RP1210_SendMessage*" to the application when EchoMode is on and will include the echo byte in the message set to 1.

Therefore, if an application is interested in reading its own messages, it can turn EchoMode to the on position.

6.1.2 J1708Mode and RP 1210B Enhancements

The default J1708Mode is in “Converted” Mode. In **Converted Mode** multiple clients (connections) can be established.

New to RP 1210B is that an RP 1210B-compliant API will allow multiple client connections in “Raw” mode. A survey taken of API vendors show that the “raw” mode restriction (of RP 1210/RP 1210A) was not really in place.

6.1.3 FilterStates

Upon *RP1210_ClientConnect*, some API/VDA vendors have **FilterStates** to “PassAll.” Vendors should initialize with FilterStates to “Discard” since some applications do not want to receive messages before the entire application is fully ready to handle the message traffic. This means that no messages should be available to the application until filter commands are issued or FilterStates are set to “PassAll.”

6.1.4 Blocking Mode

Upon *RP1210_ClientConnect*, some API/VDA vendors have “blocking mode” set to “blocking,” while others have it set to “non-blocking.” Vendors should initialize with **BlockingMode** set to “blocking” for any calls with a blocking/non-blocking parameter.

6.1.5 Blocking Timeout

New to RP 1210B is **BlockingTimeout**, a timeout value on blocked reads/writes. The default value would be infinite (as is alluded to in RP 1210A). If blocking is turned off, the API should return as soon as possible from the command (*RP1210_SendCommand*/*RP1210_ReadMessage*/*RP1210_SendMessage*).
A timeout on a call in blocking mode (with blocking not set to infinite) would return ERR_COMMAND_TIMED_OUT.

6.1.6 J1708/J1939/CAN/J1850/ISO 15765 Filter Clarifications

Users can send in more than one filter per *RP1210_SendCommand* call. This allows a programmer to send in an array of filters. Some vendors currently only acknowledge the first one. The API DLL is responsible for maintaining a table of at least 64 filters per client/protocol. Additional calls to *RP1210_SendCommand* to set a filter will add to the list. This way, the application does not have to send an updated “entire list” of filter parameters. This list is only cleared on reset or by setting all filter states to discard.

Filters can be issued that will allow J1939 address claim and J1939 transport protocol messages to be passed to the application. These filters even apply if the API is packetizing the transport protocol messages (some applications want to see both the multiple packets as well as the single transport message). Typically in the past, most APIs did not pass the address claim and transport messages to the client application. These messages do not appear until specifically asked for by setting specific filters. Therefore, “setting all filter states to pass” does not allow them to pass to the application.

6.1.7 J1708 Parameter ID (PID) Filtering

PID filtering in the API was discussed in the update committee, however it was deemed too difficult because of variable length messages, packed PIDs, etc. Any PID filtering will be done in the software application.

6.1.8 Inclusive and Exclusive Filtering

Currently, RP 1210A filtering is set to “inclusive.” This type of filtering forces the application to “ask” the API for each MID, PGN, or CANID that it wants to see. With “FilterStates=AllFilterStatesToPass”, it forces the application to handle all filtering. In RP 1210B, a new filter mechanism called “exclusive” will be added. This allow the application to say, “I want to see all messages, except from the engine, except this particular message, etc”. See *RP1210_SendCommand* for more details.

7.0 RP 1210B REQUIRED FUNCTIONS

This section describes the required functions of the RP1210 API. Note that an implementation claiming to conform with this API shall minimally support these functions without deviating in name (case-sensitivity should be preserved), type, or argument list. Implementations may exceed the minimal functionality listed here, as long as the functionality provided by the given set of functions below is rigidly maintained. The required functionality is constituted by the following set of functions:

FUNCTION NAME	BRIEF DESCRIPTION
RP1210_ClientConnect	establishes connection between the client application and the API DLL
RP1210_ClientDisconnect	disconnects the client application from the API DLL
RP1210_SendMessage	sends a message to the API DLL
RP1210_ReadMessage	reads a message from the API DLL
RP1210_SendCommand	sends a command to the API DLL
RP1210_ReadVersion	reads version information from the API
RP1210_GetErrorMsg	translates an error code into a description of the error
RP1210_GetHardwareStatus	polls the interface hardware to determine (a) if the interface hardware exists and (b) what is the datalink status.

8. CLARIFICATIONS FOR APPLICATION DEVELOPERS

8.1 UNEXPECTED VEHICLE INTERFACE ADAPTER DISCONNECT FROM VEHICLE OR COMPUTER.

There are differences in how an adapter disconnect is managed among vehicle interface adapter vendors. The RP indicates that the API should return Error Code 142, "ERR_HARDWARE_NOT_RESPONDING" to the calling application whenever a hardware error occurs. However, the reaction of vehicle interface adapter vendor APIs differ greatly when hardware is accidentally disconnected. Because of these differences, TMC recommends that all applications monitor for changes in hardware status by periodically calling the "*RP1210_GetHardwareStatus*()" function and/or processing the return code from "*RP1210_SendMessage* ()" or "*RP1210_ReadMessage* ()" to determine the status of the hardware translator. If an error condition exists, it notifies the operator and recommends that a check of the vehicle and computer connections be made before proceeding.

In severe cases the client application may be required to do the following:

1. Notify the user of the problem, ask the user to re-instate the hardware.
2. Disconnect from the API by using *RP1210_ClientDisconnect ()*.
3. Continue calling *RP1210_ClientConnect ()* until successful.
4. Set up all RP 1210 variables and filters again.
5. Reclaim all J1939 addresses (if using J1939).

Vendor APIs that require no intervention retain all client identification and filter states and will verify ownership of all previously J1939-claimed addresses upon correction of the disconnect. This type of API does not require the recovery method described above. See also the RP1210B function *RP1210_GetLastErrorMsg*.

8.2 FILTER STATE AMBIGUITIES

Differences exist among vehicle interface adapter manufacturers in their handling of filters. Setting "All Filter States to Pass" is a temporary condition that is canceled when issuing specific filter requests.

8.3 APPLICATION DEVELOPER SUMMARY

If the deployment of a software application requires compatibility with all vehicle interface adapters, then the application must be designed based upon the lowest common denominator of adapter performance. If such compatibility is not required, then users should contact the selected vehicle interface adapter manufacturer to understand the performance and functional capabilities of their products.

8.4 INI FILE VERSUS REGISTRY

During the past several years, with the operating systems specified in this document, the use of the Windows registry has become the de-facto standard way of storing variables, as opposed to the old way of using INI files. The committee recognized and discussed this, however with the many variables in-place at the moment (knowledge of user, various vendors, OEM, etc), it was decided to stick with the same format of using the INI files.

8.5 RP 1210B COMPLIANCE TESTING

During the RP1210B update process, TMC was asked to develop testing program to determine whether an API is RP 1210B compliant for the operating systems and protocols that it supports. TMC is not in a position at this time to offer such a program.

9. COMPLIANCE FOR API/VDA VENDORS

This section outlines the required features of RP 1210B for a vendor to signify RP1210B-compliance.

9.1 STANDARD FUNCTIONS

Standard commands (All commands marked must be implemented).

Function Name	**J1708/PLC**	**CAN**	**J1850**	**ISO15765**	**J1939**
RP1210_ClientConnect	X	X	X	X	X
J1708 Baud (9600 – Format 2)	X				
J1708 Baud (19200)	NO				
J1708 Baud (38400)	NO				
J1708 Baud (57600)	NO				
Multiple CAN Baud (Format 1)		NO			
Multiple CAN Baud (Format 2)		NO			
Multiple CAN Baud (Format 3)		NO			
Multiple CAN Bauds (250k - Format 4)		X			
J1850 Baud Rate (10.4k)			X		
J1850 Baud Rate (41.6k)			NO		
Multiple ISO15765 Baud				NO	
RP1210_ClientDisconnect	X	X	X	X	X
RP1210_SendMessage	X	X	X	X	X
RP1210_ReadMessage	X	X	X	X	X
RP1210_ReadVersion	X	X	X	X	X
RP1210_ReadDetailedVersion	X	X	X	X	X
RP1210_GetErrorMsg	X	X	X	X	X
RP1210_GetLastErrorMsg	X	X	X	X	X
RP1210_GetHardwareStatus	X	X	X	X	X
INI File Debug Support	NO	NO	NO	NO	NO

9.2 *RP1210_SENDCOMMAND* FUNCTIONS

The following table shows which *RP1210_SendCommand* functions must be implemented to signify TMC RP 1210B compliance.

***RP1210_SendCommand* and Parameters**

RP1210_SendCommand Parameter	J1708/PLC	CAN	J1850	ISO15765	J1939
RP1210_Reset_Device	X	X	X	X	X
RP1210_Set_All_Filters_States_to_Pass	X	X	X	X	X
RP1210_Set_Message_Filtering_For_J1939					X
RP1210_Set_Message_Filtering_For_CAN		X			
RP1210_Set_Message_Filtering_For_J1708	X				
RP1210_Set_Message_Filtering_For_J1850			X		
RP1210_Set_Message_Filtering_For_ISO15765				X	
RP1210_Generic_Driver_Command	NO	NO	NO	NO	NO
RP1210_Set_J1708_Mode	X				
RP1210_Echo_Transmitted_Messages	X	X	X	X	X
RP1210_Set_All_Filters_States_to_Discard	X	X	X	X	X
RP1210_Set_Message_Receive	X	X	X	X	X
RP1210_Protect_J1939_Address					X
RP1210_Set_Broadcast_For_J1708	X				
RP1210_Set_Broadcast_For_CAN		X			
RP1210_Set_Broadcast_For_J1939					X
RP1210_Set_Broadcast_For_J1850			X		
RP1210_Set_Broadcast_For_ISO15765				X	
RP1210_Set_BlockTimeout	X	X	X	X	X
RP1210_Set_J1708_Baud	NO				
RP1210_Set_J1708_Filter_Type	X				
RP1210_Set_J1939_Filter_Type					X
RP1210_Set_CAN_Filter_Type		X			
RP1210_Set_ISO15765_Filter_Type				X	
RP1210_Set_J1939_Interpacket_Time					NO
RP1210_Set_ISO15765_Flow_Control				X	
RP1210_Clear_ISO15765_Flow_Control				X	
RP1210_Set_ISO15765_Link_Type				X	
RP1210_Set_J1939_Baud					NO

10. RP 1210 FUNCTION CALLS

10.1 *RP1210_CLIENTCONNECT* PROTOTYPE

This function is called by the client application seeking connection with a DLL that corresponds to the implementation of this API. Inside the API DLL, the function allocates and initializes any client data structures, and loads, initializes, or activates any device drivers (virtual or otherwise) to communicate with the hardware. If the connection is successful, the function shall return a unique identifier, corresponding to the ID of the client application, as assigned by the API DLL.

10.1.1 *RP1210_ClientConnect* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_ClientConnect
(
   HWND hwndClient,
   short nDeviceID,
   const char *fpchProtocol,
   long lTxBufferSize,
   long lRcvBufferSize,
   short nIsAppPacketizingIncomingMsgs
);
```

Visual Basic

```
Declare Function RP1210_ClientConnect Lib “VENDRX32.DLL”...
(
   ByVal hwndClient As Long,
   ByVal nDeviceID As Integer,
   ByVal fpchProtocol As String,
   ByVal lTxBufferSize As Long,
   ByVal lRcvBufferSize As Long,
   ByVal nIsAppPacketizingIncomingMsgs As Integer
) As Integer
```

10.1.2 *RP1210_ClientConnect* Parameters

Parameter	Description
hwndClient	This parameter is no longer necessary and is unused (Windows 3.1).
nDeviceID	The device to which the client application is requesting connection.
fpchProtocol	Pointer to a null-terminated string of the protocol name to be used by the device designated in the previous parameter. (See **Protocol Connect String** section for protocol strings and variations.)
lTxBufferSize	A long integer for the requested size (in bytes) of the client transmit buffer to be allocated by the API for the queuing of messages sought to be transmitted by the client application. Should be passed as 0 if the application does not want to dictate the buffer size and the API DLL default of 8K is acceptable.
lRcvBufferSize	A long integer for the requested size (in bytes) of the client receive buffer to be allocated by the API for the queuing of messages meant to be received by the client application. Should be passed as 0 if the application does not want to dictate the buffer size and the API DLL default of 8K is acceptable.
nIsAppPacketizing IncomingMsgs	A flag that is relevant only for J1939. Should be set to zero if the application wants the lower-level components to perform the tasks of fragmenting the message into packets and return complete messages assembled from packets received. Should be set to 1 in the rare case where the application itself will perform the packetizing of message fragments.

10.1.3 *RP1210_ClientConnect* Return Value

If the connection is successful, then the function returns a value between 0 and 127, corresponding to the client identifier that the application program is assigned by the API DLL. The application program must save this return value in order to conduct future transactions with the DLL. If the connection is unsuccessful, then an error code is returned that corresponds to a number greater than 127. The typical error codes are described below. More descriptive or proprietary codes may be returned by individual vendor implementations as long as the numerical equivalents of these return values are greater than 192 and are clearly documented by the vendor.

Mnemonic (Return Codes)	**Description**
`ERR_CLIENT_ALREADY_CONNECTED`	A client is already connected to the specified device.
`ERR_CLIENT_AREA_FULL`	The maximum number of connections has been reached.
`ERR_CONNECT_NOT_ALLOWED`	Only one connection is allowed in the requested Mode.
`ERR_DEVICE_IN_USE`	The specified device is already in use and does not have the ability to maintain connections with multiple clients simultaneously.
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_HARDWARE_NOT_RESPONDING`	The device is not responding.
`ERR_INVALID_DEVICE`	The specified device ID is invalid.
`ERR_INVALID_PROTOCOL`	The specified protocol is invalid or unsupported or the extended connect string (fpchProtocol) was incorrect or unsupported.
`ERR_NOT_ENOUGH_MEMORY`	The API DLL could not allocate enough memory to create the client.
`ERR_J1708_BAUD_SET_NONSTANDARD`	See ***RP1210_ClientConnect* Notes**.
`ERR_J1939_BAUD_SET_NONSTANDARD`	See ***RP1210_ClientConnect* Notes**.
`ERR_CAN_BAUD_SET_NONSTANDARD`	See ***RP1210_ClientConnect* Notes**.
`ERR_MULTIPLE_CONNECTIONS_NOT_ALLOWED_NOW`	See ***RP1210_ClientConnect* Notes**.

10.1.4 RP1210_ClientConnect Notes

For an application program to call any other functions within the API DLL, the application must first receive and recognize a successful return value from this function.

If multiple J1939 applications are connected through the same device and any of them changes the *nIsAppPacketizingIncomingMsgs* flag, then the client attempting to change this flag shall receive the ERROR_CONNECTION_NOT_ALLOWED return code. In other words, as long as there exist one or more J1939 connections to a given device that have set the *nIsAppPacketizingIncomingMsgs* flag, all applications issuing this command must also set this flag in a similar fashion, or they will receive the ERROR_CONNECTION_NOT_ALLOWED.

When a J1708, J1939, CAN or ISO15765 application establishes a connection, it is assumed that no filters are set and the drivers do not allow any datalink messages to pass through. Only after filters have been set by using either command 3, 4, 5, 7 or 9 of the RP1210_SendCommand functions, can the application start receiving messages.

Applications using ISO 15765 that wish to communicate long (segmented) messages must additionally establish a flow control association between the incoming (device) CAN ID and the outgoing (tester) CAN ID with the *RP1210_Set_ISO15765_Flow_Control* command, before such messages can be sent to or received from that device. Single Frame (short) messages – such as broadcast messages - can be sent to or received from a device without recourse to the *RP1210_Set_ISO15765_Flow_Control* command.

10.1.4.1 Multiple CAN Protocols on One CAN Channel; Multiple Baud Rates (J1939/J1708)

Most VDA manufacturers only have one CAN chip in their hardware. This chip can only be initialised at one particular speed. RP1210B allows the flexibility of connecting at multiple CAN baud rates. This can create a problem where a user wants to crank up multiple CAN-based protocols with different speeds. Therefore, if an application has the baud rate on the CAN chip set at anything other than 250k (J1939, and RP1210A default), the following newly introduced error will be returned on future tries to connect to the CAN bus.

```
        ERR_CAN_BAUD_SET_NONSTANDARD (454)
```

It is recommended that when the application needs to do something like this, that it prevents other applications from connecting to the data bus through the following call:

```
        RP1210_SendCommand( RP1210_Disallow_Further_Connections )
```

Note that any further attempts to connect would then come back with the new RP1210B error code:

```
        ERR_MULTIPLE_CONNECTIONS_NOT_ALLOWED_NOW     (455)
```

This also affects J1708 and J1939 connections at non-standard baud rates.

10.1.5 Protocol Connection String

Several new protocols and features have been added to extend the functionality of *RP1210_ClientConnect*.

fpchProtocol	**Description**
CAN	CAN Network Protocol (see SETTING CAN BAUD RATE)
J1708	SAE J1708 Protocol (see SETTING J1708 BAUD RATE)
J1939	SAE J1939 Protocol
PLC	Power Line Carrier (PLC4TRUCKS) Protocol
J1850_104k	SAE J1850 Vehicle Protocol (Baud Rate of 10.4k)
J1850_416k	SAE J1850 Vehicle Protocol (Baud Rate of 41.6k)
ISO15765	ISO 15765 Vehicle Protocol (see SETTING ISO15765 BAUD RATE)

10.1.5.1 Using TCP/IP as a Transport Layer for a Protocol

Several new devices have been made available since RP 1210A was written. One such device uses TCP/IP as a transport for the protocols defined in the table above. Therefore, if a user wants to use the TCP/IP as a transport for the protocol, the user can append the following string to the fpchProtocol parameter.

```
"Transport=TCPIPaddress=xxx.xxx.xxx.xxx,TCPIPport=xx"
```

This can be used, even in addition to the following sections dealing with specialty features of the fpchProtocol parameter.

Example:
(the VDA is at address 192.198.22.33:90)

```
"J1708:Transport=TCPIPaddress=192.198.22.33,TCPIPport=90"
```

10.1.5.2 Using a Modem as a Transport Layer for a Protocol

Another transport mechanism has been provided for the RP 1210 protocols (mostly J1708). This transport mechanism is a modem. Therefore, if a user wants to use a modem as a transport for the protocol, the user can append the following string to the fpchProtocol parameter.

```
"Transport=ModemTelno=sssss"
```

This can be used, even in addition to the following sections dealing with specialty features of the fpchProtocol parameter.

Example:

```
"J1708:Transport=ModemTelno=859-923-4100"
```

10.1.5.3 SETTING CAN BAUD RATE on *RP1210_ClientConnect*

The default CAN baud rate on *RP1210_ClientConnect* is 250k (same as J1939). However, some applications may require different CAN baud rates. The following paragraphs deal with connecting at a different baud rate. For specifics on the setting of CAN baud rates, refer to SAE J1939-11 Section 3.14 "CAN Bit Timing Requirements."

There is an INI variable "CANFormatsSupported" that shows which formats that an API supports.

To correctly set the baud rate on CAN the user uses the crystal frequency of the hardware to set the following parameters:

PROP_SEG is programmable (1-8)
PHASE_SEG1 is programmable (1-8)
PHASE_SEG2 is programmable and must be < PHASE_SEG1
Resynchronization jump width is either 1 or 3
Identifier size 11 or 29

Format 1 – Driver uses the sample location to calculate PROP_SEG, PHASE_SEG1, and PHASE_SEG2

```
fpchProtocol = "CAN:Baud=X,SampleLocation=Y,SJW=Z,IDSize=S"
```

Baud (X) may be between 1 and 1M (1-1000000).
SampleLocation (Y) is between 50 and 95 (percent %)
SJW (Z) may be 1 or 3
IDSize (S) may be 11 or 29 preferred

Format 2 – Formula for baud derived directly from the BOSCH CAN specification

```
fpchProtocol="CAN:Baud=X,PROP_SEG=A,PHASE_SEG1=B,PHASE_SEG2=C,SJW=Z,IDSize=SS"
```

Baud (X) may be between 1 and 1M (1-1000000).
PROP_SEG (A) is between 1 and 8
PHASE_SEG1 is between 1 and 8
PHASE_SEG2 is must be less than PHASE_SEG1
SJW may be 1 or 3
Size may be 11 or 29

Format 3 – Formula for baud derived from Intel implementations

```
fpchProtocol="CAN:Baud=X,TSEG1=D,TSEG2=E,SJW=Z,IDSize=SS"
```

Baud (X) may be between 1 and 1M (1-1000000).
TSEG1 (D) is between 2 and 15
TSEG2 (E) is between 1 and 7 and must be less than TSEQ1
SJW may be 1 or 3
Size may be 11 or 29

Format 4 – General, Default Format (CAN baud rate defaults to 250k)

```
fpchProtocol="CAN"
```

10.1.5.4 SETTING J1708 BAUD RATE on *RP1210_ClientConnect* (fpchProtocol)

Some vendors and applications have set the J1708 datalink to values other than 9600 in an effort to speed end-of-line programming or code/parameter downloads. The following paragraphs deal with connecting at a different baud rate. There is also another way of setting the J1708 baud rate after calling *RP1210_ClientConnect*, see the *RP1210_SendCommand* documentation.

There is an INI variable "J1708FormatsSupported" that shows which formats that an API supports.

Format 1 – New Format for Variable Baud Rate

```
fpchProtocol = "J1708:Baud=X"
```

Baud (X) may be "9600", "19200", "38400", "57600"

Format 2 – General, Default Format (J1708 baud rate defaults to 9600)

```
fpchProtocol="J1708"
```

10.1.5.5 SETTING J1939 BAUD RATE on *RP1210_ClientConnect* (fpchProtocol)

Some vendors and applications have set the J1939 datalink to values other than 250k in an effort to speed end-of-line programming or code/parameter downloads. The following paragraphs deal with connecting at a different baud rate. There is also another way of setting the J1939 baud rate after calling *RP1210_ClientConnect*, see the *RP1210_SendCommand* documentation. This command was added to allow for the CM features of J1939, but at a much higher baud rate.

There is an INI variable "J1939FormatsSupported" that shows which formats that an API supports.

Format 1 – New Format for Variable Baud Rate

```
fpchProtocol = "J1939:Baud=X"
```

Baud (X) may be "125", "250", "500", "1000"

Format 2 – General, Default Format (J1939 baud rate defaults to 250k)

```
fpchProtocol="J1939"
```

10.1.5.6 SETTING ISO15765-2 BAUD RATE on *RP1210_ClientConnect*

The default ISO 15765-2 baud rate on *RP1210_ClientConnect* is 250k (same as J1939), however some applications may require different ISO15765-2 baud rates. To set the baud rate on ISO15765-2 the user can use the following values for the protocol connection string:

Format 1 – Format for Variable Baud Rate

```
fpchProtocol="ISO15765:Baud=X"
```

Baud (X) may be between 1 and 1M (1-1000000).

Format 2 – General, Default Format (ISO 15765-2 baud rate defaults to 250k)

```
fpchProtocol="ISO15765"
```

10.2 *RP1210_CLIENTDISCONNECT*

This function is called by the client application seeking to terminate its connection with the data bus to which it is connected. The API DLL will then de-allocates any client data structures and disables any active filters associated with this client. If this client was the last client connected to the API, the DLL will also deactivate any device drivers (virtual or otherwise) that communicate with the hardware.

10.2.1 *RP1210_ClientDisconnect* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_ClientDisconnect
(
      short nClientID
);
```

Visual Basic

```
Declare Function RP1210_ClientDisconnect Lib “VENDRX32.DLL”...
(
      ByVal nClientID As Integer
) As Integer
```

10.2.2 *RP1210_ClientDisconnect* Parameters

Parameter	Description
nClientID	The client identifier for the client that needs to be disconnected.

10.2.3 *RP1210_ClientDisconnect* Return Value

If the disconnect is successful, then the function returns 0. Otherwise, a value greater than 127 is returned, corresponding to the code for the error that was registered in the processing of this function call. The typical error codes are described below. More descriptive or proprietary codes may be returned by individual vendor implementations as long as the numerical equivalents of these return values are greater than 192, and the descriptions are clearly documented by the vendor.

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialized.
ERR_FREE_MEMORY	An error occurred in the process of memory deallocation.
ERR_INVALID_CLIENT_ID	The identifier for the client that is being sought to be disconnected was invalid.

10.2.4 *RP1210_ClientDisconnect* Notes

This function can only be called only after a connection has been successfully established with the API DLL. The parameter passed to this function is the identifier for the connected client that is sought to be disconnected. Typically, a call to this function will be made in the clean-up code of the client application, when it is closing. It is the responsibility of the user application to terminate all of its threads that it has created to perform the IO.

If the API DLL calls *RP1210_ClientDisconnect* while blocking on *RP1210_ReadMessage* or *RP1210_SendMessage*, the DLL will signal the blocked function(s) to return with error value ERR_CLIENT_DISCONNECTED. This method will give a client application the ability to gracefully stop any threads waiting for returns from blocked functions. Any additional disconnect information concerning thread shut down is passed back in the call to *RP1210_GetHardwareStatus*.

10.3 *RP1210_SENDMESSAGE*

This function is called by the client application to send a message, using the selected device and protocol. The message is placed on a queue and processed in the order it was received. Messages are placed in the API DLL transmit queue for processing. A message is rejected when the transmit queue becomes full or any error condition is detected. The API DLL notifies the client application through the return code of any failures.

10.3.1 *RP1210_SendMessage* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_SendMessage
(
       short nClientID,
       char *fpchClientMessage,
       short nMessageSize,
       short nNotifyStatusOnTx,
       short nBlockOnSend,
)
```

Visual Basic

```
Declare Function RP1210_SendMessage  Lib "VENDRX32.DLL"...
(
       ByVal nClientID As Integer,
       ByVal fpchClientMessage As String,
       ByVal nMessageSize As Integer,
       ByVal nNotifyStatusOnTx As Integer,
       ByVal nBlockOnSend As Integer
) As Integer
```

10.3.2 *RP1210_SendMessage* Parameters

Parameter	Description
nClientID	The client identifier for the client that wants to send a message.
fpchClientMessage	A pointer to the buffer (allocated by the client application) that contains the message to be sent. The protocol-specific message formats must be in conformance with those specified below.
nMessageSize	The size of the entire message in bytes (including any identifier, echo byte, checksum, or other qualifier bytes).
nNotifyStatusOnTx	This is a Windows 3.1 legacy parameter and will be ignored. New applications should pass in the value of zero.
nBlockOnSend	A flag to indicate whether the function must block on message transmit or not. Note that the BLOCKING TIMEOUT variable applies to blocked sends.

10.3.3 *RP1210_SendMessage* Return Value

The function will always return a value of 0 for successful queuing of the message for transmission. In the event of an error in queuing the message for transmission, an error code, corresponding to a value of greater than 127 is returned.

The typical error codes are described below. More descriptive or proprietary codes may be returned by individual vendor implementations as long as the numerical equivalents of these return values are greater than 192, and the descriptions are clearly documented by the vendor.

Mnemonic	Description
ERR_ADDRESS_LOST	The API was forced to concede the address to another node on the J1939 network. This return code is for J1939 clients only.
ERR_ADDRESS_NEVER_CLAIMED	J1939 client never issued a protect address. This return code is for J1939 clients only.
ERR_BLOCK_NOT_ALLOWED	This was a legacy Windows™ 3.1x return code.

`ERR_CLIENT_DISCONNECTED` Indicates *RP1210_ClientDisconnect* was called while blocking on *RP1210_ReadMessage* or *RP1210_SendMessage.*

`ERR_DLL_NOT_INITIALIZED` Indicates that the API DLL was not initialized.

`ERR_HARDWARE_NOT_RESPONDING` The device hardware interface, through which the message is routed, is not responding.

`ERR_INVALID_CLIENT_ID` The identifier for the client that is seeking message transmission is invalid or unrecognized.

`ERR_MESSAGE_NOT_SENT` Returned if blocking is used and the message was not sent.

`ERR_MESSAGE_TOO_LONG` The message being sought to be transmitted is too long.

`ERR_TX_QUEUE_CORRUPT` The API DLL's receive message queue is corrupt.

`ERR_TX_QUEUE_FULL` The API DLL's transmit message queue is full.

10.3.4 *RP1210_SendMessage* Message Definitions (fpchClientMessage)

The message construction for the different protocols supported by this API must be in conformance with the specification detailed below.

10.3.5 Formatting a J1708 Message for *RP1210_SendMessage*

J1708

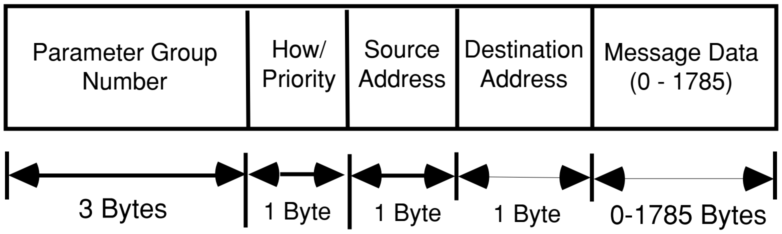
Priority	MID **	Message Data	Checksum**
1	0/1	1-504/505 bytes	0/1

The *nMessageSize* parameter should reflect the size of the entire message, including the message priority byte and checksum (if in RAW MODE).

Parameter	**Description**
Priority	One byte identifying what J1708 priority the message should be transmitted at.
MID**	In J1587 communications, this is the J1587 Message Identifier (MID). Otherwise, in J1708, this field is omitted.
Message Data	This field includes all message data and may be between 1 and 505 bytes.
Checksum**	If the API is in RAW MODE, then the application would have to add the checksum and the *nMessageSize* parameter would be incremented by one to show the checksum byte. It is the responsibility of the API to add a checksum byte to the message before sending the J1708 message to the datalink if the API DLL is in *CONVERTED MODE*.

10.3.6 Formatting a J1939 Message for *RP1210_SendMessage*

J1939



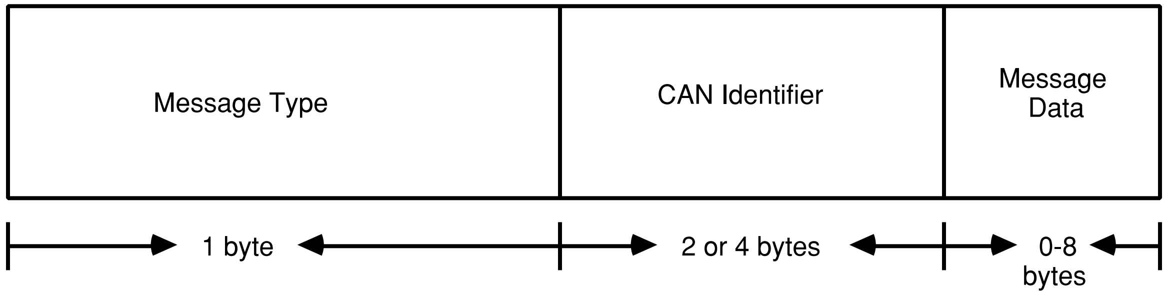
The *nMessageSize* parameter should reflect the size of the entire message.

The J1939 message is constructed from five fields:

Parameter	Description
Parameter Group No.	Three bytes in length and is in little endian/Intel format.
How To Send/Priority	A bit field, one byte (eight bits) in length.
	• The higher order bit, bit 7, determines the J1939 transport type to use for data field lengths greater than eight bytes. A value of one in the high order bit will cause the lower level driver to use the Broadcast Announcement Message (BAM) transport, a value of zero will enable the Connection Management (RTS/CTS) transport facilities (reference SAE J1939/21). If this bit is not set (indicating RTS/CTS) and the global address is used, then BAM will be used as transport (this is a clarification from RP 1210A). • The lower three bits (bits 0-2) determine the message priority. • The other bits (bits 3-6) are reserved for future use and should be set to zero.
Source Address	One byte in length and contains the originating nodes address. If the Source Address field contains an address that has not been claimed, the API will still send the message (see Section A3.5, Protect J1939 Address).
Destination Address	One byte in length and contains the address of the node for which the packet is bound.
Data	Contains all the data elements of the message. The API DLL, or its associated drivers, is required to break down long messages and transmit multi-packet message fragments in accordance with the J1939 transport protocol detailed in Section 3.10 of SAE J1939/21.

10.3.7 Formatting a CAN Message for *RP1210_SendMessage*

CAN



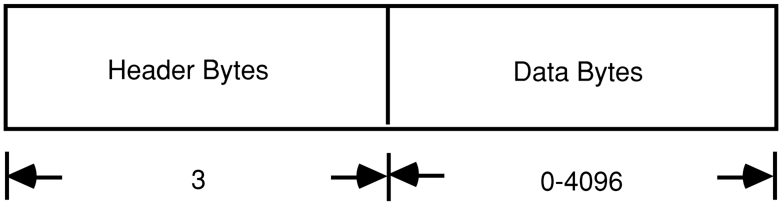
The *nMessageSize* parameter for a CAN transmit message should include the byte count for the message type as well as the CAN Identifier.

The CAN message is constructed from five fields:

Parameter	Description
Message Type	Standard CAN message = 0x00 (11-bit identifier) Extended CAN message = 0x01 (29-bit identifier)
CAN Identifier (CAN ID)	CAN ID for Standard CAN message is two bytes (big endian format). CAN ID for Extended CAN message is four bytes (big endian format).
Message Data	Message data from 0 to 8 bytes.

10.3.8 Formatting a J1850 Message for *RP1210_SendMessage*

J1850



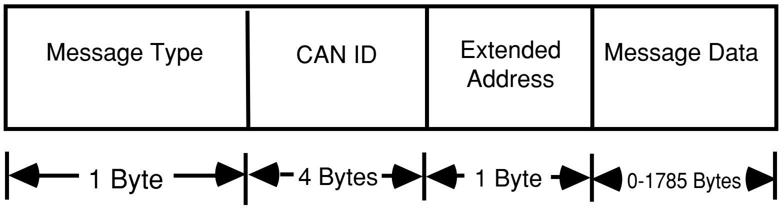
The *nMessageSize* parameter for a J1850 transmit message should include the byte count for the entire message.

The message is constructed from the following fields:

Parameter	Description
Header Bytes	3 Bytes for the message header.
Data Bytes	From 0-4096 data bytes (12 bytes max in “Normal Mode” and 4096 bytes max in “Block Mode”.

10.3.9 Formatting an ISO 15765 Message for *RP1210_SendMessage*

ISO15765



The *nMessageSize* parameter for an ISO 15765 transmit message should also include the byte count for the message type, CAN Identifier and extended address byte.

Before sending long (segmented) messages, the application must:

a. Configure the filtering such that messages will be accepted from the incoming (device) CAN-ID with either the *RP1210_Set_Message_Filtering_For_ISO15765* or the *RP1210_Set_All_Filters_States_to_Pass* command, and
b. Establish a flow control association between the incoming (device) CAN ID and the outgoing (tester) CAN ID with the *RP1210_Set_ISO15765_Flow_Control* command.

The API DLL, or its associated drivers, is required to break down long messages and communicate multi-packet message fragments in accordance with the protocol detailed in ISO 15765-2.

It is not necessary to establish a flow control association to be able to send short (Single Frame) messages to a device.

The ISO 15765 message is constructed from four fields:

Parameter	**Description**
Message Type	Standard (11-bit) CAN message = STANDARD_CAN (0x00) Extended (29-bit) CAN message = EXTENDED_CAN (0x01) Standard (11-bit) CAN message with ISO15765 extended address byte = STANDARD_CAN_ISO15765_EXTENDED (0x02) Extended (29-bit) CAN message with ISO15765 extended address byte = EXTENDED_CAN_ISO15765_EXTENDED (0x03) Mixed addressing (11-bit) CAN message = STANDARD_MIXED_CAN_ISO15765 (0x04)
CAN Identifier (CAN ID)	Byte order is big endian. To stay generic, four bytes are used for the CAN identifier regardless of whether Standard or Extended CAN is used. In the case of Standard CAN, only the low (last) two bytes would be used.
Extended address byte	If the message type is STANDARD_CAN_ISO15765_EXTENDED (0x02) or EXTENDED_CAN_ISO15765_EXTENDED (0x03), this byte will specify the ISO15765 extended address byte. If the message type is STANDARD_MIXED_CAN_ISO15765 this byte will specify the ISO15765 network address extension. Otherwise, this field must be present, but is ignored.
Message Data	Message data from 0 to 4096 bytes.

10.4 RP1210_*READMESSAGE*

This function is called by the client application to read a message from the device and protocol associated with it.

10.4.1 *RP1210_ReadMessage* Prototype

C/C++

```
--declpsec (dll export) WINAPI RP1210_ReadMessage
(
       short nClientID,
       char far* fpchAPIMessage,
       short nBufferSize,
       short nBlockOnRead
)
```

Visual Basic Prototype

```
Declare Function RP1210_ReadMessage  Lib “VENDRX32.DLL”
(
       ByVal nClientID As Integer,
       ByVal fpchAPIMessage As String
       ByVal nBufferSize As Integer,
       ByVal nBlockOnRead As Integer
) As Integer
```

10.4.2. *RP1210_ReadMessage* Parameters

Parameter	Description
nClientID	The client identifier for the client that wants to send a message.
fpchAPIMessage	A pointer to the buffer (allocated by the client application) to where the transfer of the message is sought. If a message is correctly received, then the first four bytes of the message represents the timestamp of the message (the resolution of this timestamp is implementation-specific, and can be retrieved from the vendor-supplied INI file), and the bytes following the timestamp shall be protocol-specific in accordance with the protocol sections detailed below.
nBufferSize	The size of the message in bytes (including any identifier or other qualifier bytes preceding the actual message data).
nBlockOnRead	A flag to indicate whether the API should block on message read. Note that the BLOCKING TIMEOUT variable applies to blocked reads.

10.4.3 *RP1210_ReadMessage* Return Value

If the read is successful, then the function returns the number of bytes read including the four bytes for timestamp. If no message is present, then the function returns 0. If an error occurred, then a value, corresponding to the additive inverse of the error code, is returned. The typical error codes are described below. More descriptive or proprietary codes may be returned by individual vendor implementations as long as the numerical equivalents of these return values are greater than 192, and the descriptions are clearly documented by the vendor.

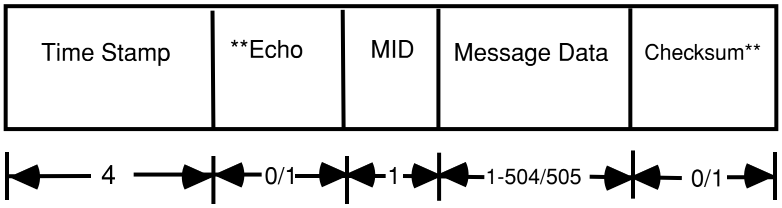
Return Codes

Mnemonic	Description
`ERR_CLIENT_DISCONNECTED`	Indicates *RP1210_ClientDisconnect* was called while blocking on *RP1210_ReadMessage* or *RP1210_SendMessage.*
`ERR_DLL_NOT_INITIALIZED`	Indicates that the API DLL was not initialized.
`ERR_HARDWARE_NOT_RESPONDING`	The device hardware interface, through which the message is routed, is not responding.
`ERR_INVALID_CLIENT_ID`	The identifier for the client that is seeking message transmission is invalid or unrecognized.
`ERR_MESSAGE_TOO_LONG`	The message being sought to be received is too long for the user’s buffer.
`ERR_RX_QUEUE_CORRUPT`	The API DLL’s receive message queue is corrupt.
`ERR_RX_QUEUE_FULL`	The API DLL’s receive message queue is full.

10.5 *RP1210_READMESSAGE* MESSAGE DEFINITIONS

The message reception for the different protocols supported by this API must be in conformance with the specification detailed below.

J1708



Time Stamp	**Echo	MID	Message Data	Checksum**
4	0/1	1	1-504/505	0/1

10.5.1 The J1708 (and PLC4TRUCKS) Message Returned From *RP1210_ReadMessage*

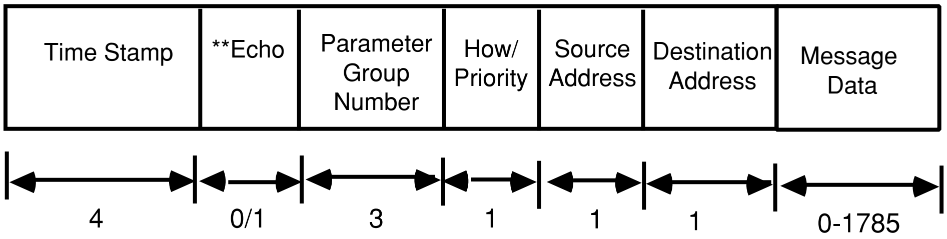
If the message is correctly received, the return value from *RP1210_ReadMessage* will specify the size of the entire message, including the four timestamp bytes, the entire message; echo and checksum bytes (if applicable**). This field has a maximum size of 511 bytes.

Parameter	**Description**
Timestamp	The timestamp is four bytes in length and is in big endian or Motorola format. The resolution of the timestamp is implementation-dependent and is given in the INI file.
Echo Byte**	If the command *Set Echo Transmitted Messages* has configured the API to echo transmitted messages, this field is set to 0x01 if the received message originated at the API's client application. If the received message did not originate at the client application, this byte is set to 0x00. If ECHO MODE is off, this field will not appear in the message.
Data	This is the actual message in accordance with the J1708 (or J1587) standard.
Checksum**	If the API is in RAW MODE, then the application would receive a checksum byte. Note: If there is a data bus collision and both senders back off on the first character, then the message will only be one byte long. The one byte goes in Message Data field, so there is a special case in Raw mode where there is no checksum byte.

10.5.2 The J1939 Message Returned From *RP1210_ReadMessage*

If the message is correctly received, the return value from *RP1210_ReadMessage* will specify the size of the entire message, including the four timestamp bytes, the entire message; and echo byte (if applicable**). This field has a maximum size of 1796 bytes.

J1939



Time Stamp	**Echo	Parameter Group Number	How/ Priority	Source Address	Destination Address	Message Data
4	0/1	3	1	1	1	0-1785

The J1939 message is constructed from seven fields:

Parameter	Description
Timestamp	The timestamp is four bytes in length and is in big endian or Motorola format. The resolution of the timestamp is implementation-dependent and is given in the INI file. For Messages greater that eight bytes, the TimeStamp field will contain the time that the last packet is received.
Echo Byte**	If the command *Set Echo Transmitted Messages* has configured the API to echo transmitted messages, this field is set to 0x01 if the received message originated at the API's client application. If the received message did not originate at the client application, this byte is set to 0x00. If ECHO MODE is off, this field will not appear in the message.
Parameter Group Number (PGN)	Three bytes in length and is in little endian/Intel format.
How To Send/Priority	The higher order bit 7 determines the J1939 transport type that was used to transport the data if the message is greater than eight bytes. A value of one in the high order bit, shows that BAM was used and a value of zero shows that RTS/CTS was used. The lower three bits (bits 0-2) determine the message priority. The other bits (bits 3-6) are unused and should be set to zero.
Source Address	One byte in length and contains the origination address of the message.
Destination Address	One byte in length and contains the destination address of the message.
Data	Contains all the data elements of the message.

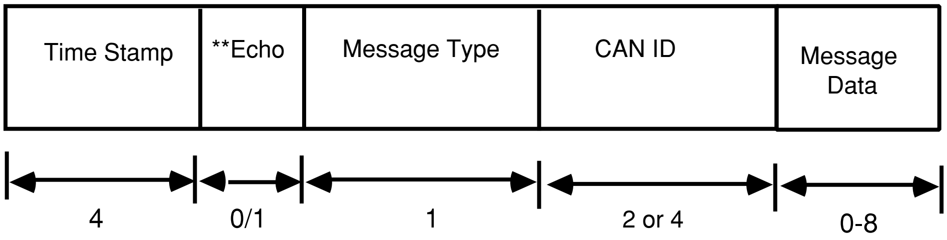
Application Notes:
The forthcoming J1939-21 is going to be using the Data Page bit and the Reserved bit to allow ISO 15765 messages to be transported across a J1939 bus. Current RP 1210A drivers let these ISO 15765 messages come through because nobody was checking the reserved bit.

TMC cannot change the field format without breaking backwards compatibility so TMC added that the vendor's driver needs to examine those bits and stop the ISO 15765 messages. The new J1939-21 is under ballot by SAE and should be approved by the end of the year 2005.

10.5.3 The CAN Message Returned From *RP1210_ReadMessage*

If the message is correctly received, the return value from *RP1210_ReadMessage* will specify the size of the entire message, including the four timestamp bytes, the entire message; and echo byte (if applicable**). This field has a maximum size of 18 bytes.

CAN



The CAN message is constructed from five fields:

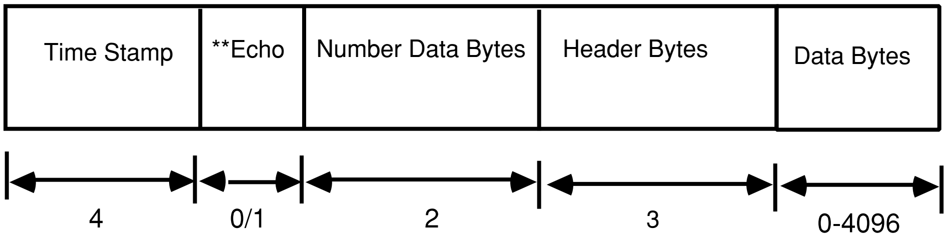
Parameter	Description
Timestamp	The timestamp is four bytes in length and is in big endian or Motorola format. The resolution of the timestamp is implementation-dependent and is given in the INI file.
Echo Byte**	If the command Set *Echo Transmitted Messages* has configured the API to echo transmitted messages, this field is set to 0x01 if the received message originated at the API's client application. If the received message did not originate at the client

	application, this byte is set to 0x00. If ECHO MODE is off, this field will not appear in the message.
Message Type	Standard CAN message = 0x00 (11-bit identifier) Extended CAN message = 0x01 (29-bit identifier)
CAN Identifier (CAN ID)	CAN ID for Standard CAN message is two bytes (big endian format). CAN ID for Extended CAN message is four bytes (big endian format).
Message Data	Message data from 0 to 8 bytes.

10.5.4 The J1850 Message Returned From *RP1210_ReadMessage*

If the message is correctly received, the return value from *RP1210_ReadMessage* will specify the size of the entire message.

J1850



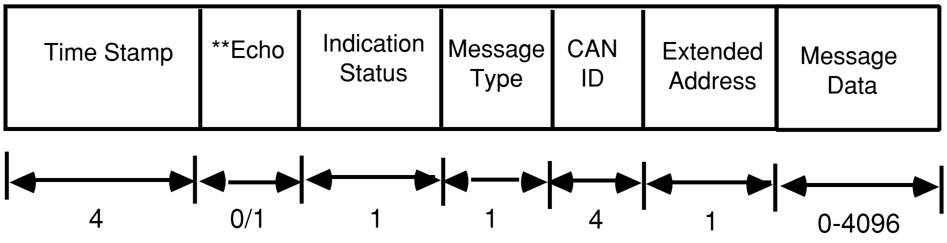
The J1850 message is constructed from the following fields:

Parameter	Description
Timestamp	The timestamp is four bytes in length and is in big endian or Motorola format. The resolution of the timestamp is implementation-dependent and is given in the INI file.
Echo Byte**	If the command Set Echo Transmitted Messages has configured the API to echo transmitted messages, this field is set to 0x01 if the received message originated at the API's client application. If the received message did not originate at the client application, this byte is set to 0x00. If ECHO MODE is off, this field will not appear in the message.
Number of Data Bytes	The number of data bytes that will appear in the message data. (in Big Endian or Motorola format)
Header Bytes	Message header, 3-bytes.
Data Bytes	Message data, from 0-4096 bytes. "Normal Mode" in J1850 allows for 12 bytes of data and "Block Mode" allows up to 4096 bytes.

10.5.5 The ISO15765 Message Returned From *RP1210_ReadMessage*

If the message is correctly received, the return value from *RP1210_ReadMessage* will specify the size of the entire message, including the four timestamp bytes, the entire message, the ISO15765 indication status byte, the CAN ID, the extended address byte, and echo byte (if applicable**). This field has a maximum size of 4108 bytes.

ISO 15765



Long (segmented) messages can only be received from a given device if the application has:

- Configured the filtering such that messages will be accepted from the incoming (device) CAN-ID with either the *RP1210_Set_Message_Filtering_For_ISO15765* or the *RP1210_Set_All_Filters_States_to_Pass* commands, and;
- Established a flow control association between the incoming (device) CAN ID and the outgoing (tester) CAN ID with the *RP1210_Set_ISO15765_Flow_Control* command.

The API DLL, or its associated drivers, is required to construct long messages and communicate multi-packet message fragments in accordance with the protocol detailed in ISO15765-2.

It is not necessary to establish a flow control association to be able to receive short (Single Frame) messages from a device.

The ISO15765 message is constructed from seven fields:

Parameter	**Description**
Timestamp	The timestamp is four bytes in length and is in big endian or Motorola format. The resolution of the timestamp is implementation-dependent and is given in the INI file. For complete segmented messages, the TimeStamp field will contain the time that the last packet is received.
Echo Byte**	If the command *Set Echo Transmitted Messages* has configured the API to echo transmitted messages, this field is set to 0x01 if the received message originated at the API's client application. If the received message did not originate at the client application, this byte is set to 0x00. If ECHO MODE is off, this field will not appear in the message.
ISO15765 Indication Status	The ISO15765 indication status of the message. • Actual message data = ISO15765_ACTUAL_MESSAGE (0x00) • ISO15765 Confirm = ISO15765_CONFIRM (0x01). This indicates that a single frame message or the last frame of a multi-frame segmented message has been transmitted, or, if the message failed to transmit, the error that occurred. (see ISO15765-2 N_USData.confirm). • ISO15765 FF (First Frame) indication = ISO15765_FF_INDICATION (0x02). This indicates that the first frame of a multi-frame segmented message has been received. (see ISO15765-2 N_USData_FF.indication).
Message Type	Standard (11-bit) CAN message = STANDARD_CAN (0x00) Extended (29-bit) CAN message = EXTENDED_CAN (0x01) Standard (11-bit) CAN message with ISO15765 extended address byte = STANDARD_CAN_ISO15765_EXTENDED (0x02) Extended (29-bit) CAN message with ISO15765 extended address byte = EXTENDED_CAN_ISO15765_EXTENDED (0x03) Mixed addressing (11-bit) CAN message = STANDARD_MIXED_CAN_ISO15765 (0x04)
CAN Identifier (CAN ID)	Byte order is big endian. To stay generic, four bytes are used for the CAN identifier regardless of whether Standard or Extended CAN is used. In the case of Standard CAN, only the low (last) two bytes would be used.
Extended Address Byte	If the message type is STANDARD_CAN_ISO15765_EXTENDED (0x02) or EXTENDED_CAN_ISO15765_EXTENDED (0x03), this byte will specify the ISO15765 extended address byte. If the message type is STANDARD_MIXED_CAN_ISO15765 this byte will specify the ISO15765 network address extension. Otherwise, this field will be present, but must be ignored.

Message Data	Message data from 0 to 4096 bytes. Actual message data is only available when the ISO15765 Indication byte is set to zero. When the ISO15765 Indication byte is set to 0x01 (“confirm”), the first byte of the message data will contain the result of the transmission (N_Result). This is as defined in ISO15765-2: • N_OK = 0x00 • N_TIMEOUT_A = 0x01 • N_TIMEOUT_Bs = 0x02 • N_INVALID_FS = 0x05 • N_WFT_OVRN = 0x07 • N_BUFFER_OVERFLOW = 0x08 • N_ERROR = 0x09

10.6 *RP1210_READVERSION*

This function is called by the client application to read the version information for the API DLL. This function has been superseded by *RP1210_ReadDetailedVersion*, although it will remain for backwards compatibility.

10.6.1 *RP1210_ReadVersion* Prototype

C/C++

```
void __declspec (dll export) WINAPI RP1210_ReadVersion
(
       char *fpchDLLMajorVersion,
       char *fpchDLLMinorVersion,
       char *fpchAPIMajorVersion,
       char *fpchAPIMinorVersion
);
```

Visual Basic

```
Declare Sub RP1210_ReadVersion Lib “VENDRX32.DLL”
(
       ByVal fpchDLLMajorVersion As String,
       ByVal fpchDLLMinorVersion As String,
       ByVal fpchAPIMajorVersion As String,
       ByVal fpchAPIMinorVersion As String
)
```

10.6.2 *RP1210_ ReadVersion* Parameters

Parameter	**Description**
fpchDLLMajorVersion	A pointer to a character (a string in Visual Basic), which the API DLL fills with the major version of the DLL that corresponds to its implementation.
fpchDLLMinorVersion	A pointer to a character (a string in Visual Basic), which the API DLL fills with the minor version of the DLL that corresponds to its implementation.
fpchAPIMajorVersion	A pointer to a character (a string in Visual Basic), that corresponds to the major version of the TMC-produced API document with which the DLL conforms. This version’s ACSII character is “2.”
fpchAPIMinorVersion	A pointer to a character (a string in Visual Basic), that corresponds to the minor version of the TMC-produced API document with which the DLL conforms. This version’s ACSII character is “0.”

10.6.3 RP1210_ ReadVersion Return Value

This call is listed as a subroutine and does not return a value.

10.7 *RP1210_READDETAILEDVERSION*

This function is called by the client application seeking to get detailed information about the vendor API and/or firmware of the device to which it is connected. This function returns an API vendor-specific string that is oftentimes helpful when debugging an application or a field installation problem. This is a new function to RP 1210B because *RP1210_ReadVersion* did not provide information about the firmware that an adapter may have. Developers should call this function instead of the *RP1210_ReadVersion* because it provides more data about the API/DLL and firmware. If an API is requested to process this request and does not have firmware associated with the device, the *fpchFWVersionInfo* parameter will be returned as a NULL string.

For example, a vendor may need to know more than just the major/minor information of their API. A couple of examples are: “2.5.02”, “2.5”, “3.22.A.109b” and “” (the NULL string).

10.7.1 *RP1210_ReadDetailedVersion* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_ReadDetailedVersion
(
       short  nClientID,
       char *fpchAPIVersionInfo,
       char *fpchDLLVersionInfo,
       char *fpchFWVersionInfo
);
```

Visual Basic

```
Declare Function RP1210_ReadDetailedVersion Lib “VENDRX32.DLL”...
(
       ByVal  nClientID As Integer,
       ByVal fpchAPIVersionInfo as String,
       ByVal fpchDLLVersionInfo as String,
       ByVal fpchFWVersionInfo as String,
) As Integer
```

10.7.2 *RP1210_ ReadDetailedVersion* Parameters

Parameter	Description
nClientID	The client identifier for the client that wants hardware status info.
fpchAPIVersionInfo	Pointer to a buffer that is 17 bytes long. The API may return up to 16 bytes in this field (with NULL terminator) to describe the version of the API interface.
fpchDLLVersionInfo	Pointer to a buffer that is 17 bytes long. The API may return up to 16 bytes in this field (with NULL terminator) to describe the version of the DLL.
fpchFWVersionInfo	Pointer to a buffer that is 17 bytes long. The API may return up to 16 bytes in this field (with NULL terminator) to describe the version of the firmware that is present in the device. A NULL string is also a valid return if the VDA does not have associated firmware.

10.7.3 *RP1210_ ReadDetailedVersion* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialised.
ERR_HARDWARE_NOT_RESPONDING	The device hardware interface, through which the message is routed, is not responding.
ERR_INVALID_CLIENT_ID	The identifier for the client that is being sought to be disconnected was invalid.

10.8 *RP1210_GETERRORMSG*

This function is called by the client application to get a textual representation of the last error code that occurred in the application.

10.8.1 *RP1210_GetErrorMsg* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_GetErrorMsg
(
short ErrorCode,
char *fpchDescription
)
```

Visual Basic

```
Declare Function RP1210_GetErrorMsg Lib "VENDRX32.DLL"
(
       ByVal ErrorCode As Integer,
       ByVal fpchDescription as String
) As Integer
```

10.8.2 *RP1210_GetErrorMsg* Parameters

Parameter	Description
ErrorCode	Numerical value for the last error which occurred.
fpchDescription	A pointer to the buffer (allocated by the client application) of 80 bytes, used to return the error message associated with the error code. The returned string length should be no greater than 80 bytes in length including the NULL character (79 bytes of message).

10.8.3 RP1210_GetErrorMsg Return Value

If the function was able to successfully convert the given error code to a useful description, a value of 0 will be returned. If the function could not convert the error into a description an error code will be returned.

Mnemonic	Description
ERR_CODE_NOT_FOUND	The error code provided does not have an associated textual representation associated with it.

10.9 *RP1210_GETLASTERRORMSG*

New to RP 1210B, this function is called by the client application to get a more descriptive subordinate error code and textual representation of the last error code. The application/thread/process passes the last error code received (in the *ErrorCode* parameter) and the function returns a more descriptive subordinate error code (in the *SubErrorCode* parameter) along with a textual message that can be delivered to the end-user (in the *fpchDescription* parameter).

- This function is not intended to supersede the *RP1210_GetErrorMsg* function. *RP1210_GetErrorMsg* can be called at any time, whereas this function is intended to be called immediately after a failed API call.
- The subordinate error code should be a #define in the VDA API's header file, with a meaningful comment.
- For more information on why this function was added, see the section entitled "Error Code 142 "ERR HARDWARE NOT RESPONDING" and Backwards Compatibility."

10.9.1 *RP1210_GetLastErrorMsg* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_GetLastErrorMsg
(
short ErrorCode,
int *SubErrorCode,
char *fpchDescription
)
```

Visual Basic

```
Declare Function RP1210_GetLastErrorMsg Lib "VENDRX32.DLL"
(
       ByVal ErrorCode As Integer,
       ByRef SubErrorCode As Integer,
       ByVal fpchDescription as String
) As Integer
```

10.9.2 *RP1210_GetLastErrorMsg* Parameters

Parameter	Description
ErrorCode	Numerical value for the last error which occurred.
SubErrorCode	A pointer to a 32-bit integer that will receive the subordinate error code description, or -1 if no subordinate error code is available.
fpchDescription	A pointer to a buffer (allocated by the client application) of 80 bytes that is used to return a detailed error message associated with the error code (and subordinate code if available). The returned string length should be no greater than 79 bytes in length and will have a NULL (0x00) terminator.

10.9.3 *RP1210_GetLastErrorMsg* Return Value

If the function was able to successfully convert the given error code to a useful description, a value of 0 will be returned. If the function could not convert the error into a description an error code will be returned.

Mnemonic	Description
`ERR_CODE_NOT_FOUND`	The error code provided does not have an associated textual representation associated with it.

10.10 *RP1210_GETHARDWARESTATUS*

This function is called by the client application to determine the hardware interface status and whether the VDA device is physically connected or not.

10.10.1 *RP1210_GetHardwareStatus* Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_GetHardwareStatus
(
       short  nClientID,
       char  *fpchClientInfo,
       short  nInfoSize,
       short  nBlockOnRequest
);
```

Visual Basic

```
Declare Function RP1210_GetHardwareStatus Lib “VENDRX32.DLL”
(
       ByVal nClientID As Integer,
       ByVal fpchClientInfo As String,
       ByVal nInfoSize As Integer
) As Integer
```

10.10.2 *RP1210_GetHardwareStatus* Parameters

Parameter	Description
nClientID	The client identifier for the client that wants hardware status info.
fpchClientInfo	A pointer to the buffer (allocated by the client application) where hardware status information is to be placed. The format of this buffer is defined as follows: 9 pairs of bytes. The low byte of each pair represents the status of the hardware/protocol. The high byte of each pair indicates the number of clients currently connected to the hardware/protocol. So, with the HWS or Hardware Status bytes, the low byte indicates whether the device is present. The high byte is the number of active connections. The rest indicate protocol information. The low byte indicates whether the protocol is connected and the high indicates the number of connected clients under that protocol. See below for information on the return.
nInfoSize	Always set to 18 bytes. API vendors should look at this value and only return the number of bytes that an application requested (since a smaller value may indicate an application was written for RP1210-0 or RP1210-A).
nBlockOnRequest	A flag to indicate whether the function must block on requesting the hardware status or not.

10.10.3 Information Returned in *fpchClientInfo*

RP1210 GetHardwareStatus

HW Status	J1939	J1708/PLC	CAN	J1850	Vendor 1	Vendor 2	Vendor 3	ISO 15765
0/1	2/3	4/5	6/7	8/9	10/11	12/13	14/15	16/17

10.10.3.1 Hardware Status (Bytes 0-1)

Byte 0

Bit 0 is set if the hardware device has been located.

Bit 1 is set if the located hardware is an internal device.

Bit 2 is set if the located hardware is an external device.

Bit 3 reserved for future RP1210 use (set to 0).

Bits 4-7 are available for vendor specific use.

Byte 1

Indicates the current number of clients connected to this device.

10.10.3.2 J1939 (Bytes 2-3)

Byte 2

Bit 0 is set if the J1939 link has been activated.

Bit 1 is set if traffic has been detected on this link within the last 1 second.

Bit 2 is set if CAN controller reports a BUS_OFF status.

Bit 3 reserved for future RP1210 use (set to 0).

Bits 4-7 are available for vendor specific use.

Byte 3

Indicates the current number of clients connected to this link.

10.10.3.3 J1708 (Bytes 4-5)

Byte 4

Bit 0 is set if the J1708 link has been activated.

Bit 1 is set if traffic has been detected on this link within the last 1 second.

Bits 2-3 reserved for future RP1210 use (set to 0).

Bits 4-7 are available for vendor specific use.

Byte 5

Indicates the current number of clients connected to this link.

10.10.3.4 CAN (Bytes 6-7)

Byte 6

Bit 0 is set if the CAN link has been activated.

Bit 1 is set if traffic has been detected on this link within the last 1 second.

Bit 2 is set if CAN controller reports a BUS_OFF status.

Bit 3 reserved for future RP1210 use (set to 0).

Bits 4-7 are available for vendor specific use.

Byte 7

Indicates the current number of clients connected to this link.

10.10.3.5 J1850 (Bytes 8-9)

Byte 8

Bit 0 is set if the J1850 link has been activated.
Bit 1 is set if traffic has been detected on this link within the last 1 second.
Bit 2
Bit 3
Bits 4-7 are available for vendor specific use.

Byte 9

Indicates the current number of clients connected to this link.

10.10.3.6 Vendor 1, 2, 3 (Bytes 10/11, 12/13, 14/15)

These bytes are all vendor specific. Information about the data format for them can be obtained from the vendor of the VDA.

10.10.3.7 ISO 15765 (Bytes 16-17)

Byte 16

Bit 0 is set if the ISO15765 link has been activated.
Bit 1 is set if traffic has been detected on this link within the last 1 second.
Bit 2 is set if CAN controller reports a BUS_OFF status.
Bit 3 reserved for future RP1210 use (set to 0).
Bits 4-7 are available for vendor specific use.

Byte 17

Indicates the current number of clients connected to this link.

10.10.4 *RP1210_GetHardwareStatus* Return Value

If the function determines that the hardware interface is present and functioning, then the function returns a value of 0. Otherwise, an error code, corresponding to a value of greater than 127 is returned. If no error occurs, the function also places datalink information in the client application provided buffer.

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_CLIENT_DISCONNECTED`	Indicates *RP1210_ClientDisconnect* was called while blocking on *RP1210_GetHardwareStatus*.
`ERR_INVALID_CLIENT_ID`	Indicates that the identifier for the client that is requesting the hardware status is invalid.

10.11 *RP1210_SENDCOMMAND*

This function is called by the client application to send a command to the device driver(s) associated with it. The command is typically intercepted by the driver and processed according to the command number. The definition of the *fpchClientCommand* character buffer is dependent upon the nCommandNumber parameter.

If these commands are successful, then the function returns a value of 0 (unless otherwise specified for a specific command). Otherwise, an error code, corresponding to a value of greater than 127 is returned. The specific error codes for each function are described in each section.

10.11.1 RP1210_SendCommand Prototype

C/C++

```
short __declspec (dll export) WINAPI RP1210_SendCommand
(
       short nCommandNumber,
       short nClientID,
       char *fpchClientCommand,
       short nMessageSize
);
```

Visual Basic

```
Declare Function RP1210_SendCommand Lib "VENDRX32.DLL"
(
       ByVal nCommandNumber As Integer,
       ByVal nClientID As Integer,
       ByVal fpchClientCommand As String,
       ByVal nMessageSize As Integer
) As Integer
```

10.11.2 *RP1210_ SendCommand* Parameters

Parameter	Description
nCommandNumber	The command number, specified below for different functions defined as part of the TMC standard. Proprietary commands should be greater than 255 (but not equal to 305 which is the "Set J1708 Baud" command.
nClientID	The client identifier for the client that wants to transmit the command to its corresponding driver(s).
fpchClientCommand	A pointer to the buffer (allocated by the client application). This buffer is defined based on the command number.
nMessageSize	The total number of bytes in the fpchClientCommand buffer.

10.11.3 Values for the *nCommandNumber* Parameter

TMC has reserved some values for the *nCommandNumber* parameter to support functionality considered required functionality. This section defines, in detail, the structure and constitution of these commands. The table below summarizes the send commands defined in this RP.

Command	Mnemonic (Defined in API Header File)	Value
Reset Device	RP1210_Reset_Device	0
Set All Filter States to Pass	RP1210_Set_All_Filters_States_to_Pass	3
Set Message Filtering for J1939	RP1210_Set_Message_Filtering_For_J1939	4
Set Message Filtering for CAN	RP1210_Set_Message_Filtering_For_CAN	5
Set Message Filtering for J1708	RP1210_Set_Message_Filtering_For_J1708	7
Set Message Filtering for J1850	RP1210_Set_Message_Filtering_For_J1850	8
Set Message Filtering for ISO15765	RP1210_Set_Message_Filtering_For_ISO15765	9
Generic Driver Command	RP1210_Generic_Driver_Command	14
Set J1708 Mode	RP1210_Set_J1708_Mode	15
Set Echo Transmitted Messages	RP1210_Echo_Transmitted_Messages	16
Set All Filter States to Discard	RP1210_Set_All_Filters_States_to_Discard	17
Set Message Receive	RP1210_Set_Message_Receive	18
Protect J1939 Address	RP1210_Protect_J1939_Address	19
Set Broadcast List for J1708	RP1210_Set_Broadcast_For_J1708	20
Set Broadcast List for CAN	RP1210_Set_Broadcast_For_CAN	21
Set Broadcast List for J1939	RP1210_Set_Broadcast_For_J1939	22
Set Broadcast List for J1850	RP1210_Set_Broadcast_For_J1850	23
Set J1708 Filter Type	RP1210_Set_J1708_Filter_Type	24
Set J1939 Filter Type	RP1210_Set_J1939_Filter_Type	25
Set CAN Filter Type	RP1210_Set_CAN_Filter_Type	26
Set J1939 Broadcast Interpacket Timing	RP1210_Set_J1939_Interpacket_Time	27
Set Max Error Message Return Size	RP1210_SetMaxErrorMsgSize	28
Disallow Further Client Connections	RP1210_Disallow_Further_Connections	29
Set J1850 Filter Type	RP1210_Set_J1850_Filter_Type	30
Release a J1939 Address	RP1210_Release_J1939_Address	31
Set ISO15765 Filter Type	RP1210_Set_ISO15765_Filter_Type	32
Set Broadcast List for ISO15765	RP1210_Set_Broadcast_For_ISO15765	33
Set ISO15765 Flow Control	RP1210_Set_ISO15765_Flow_Control	34
Clear ISO15765 Flow Control	RP1210_Clear_ISO15765_Flow_Control	35
Set ISO15765 Link Type.	RP1210_Set_ISO15765_Link_Type	36
Set J1939 Baud Rate	RP1210_Set_J1939_Baud	37
Set Blocking Timeout	RP1210_Set_BlockTimeout	215
Set J1708 Baud Rate	RP1210_Set_J1708_Baud	305

10.11.4 *RP1210_ Reset_Device*

This API command allows the user to attempt to reset the physical device. This command is allowed only when one client is actively connected. If more than one client is connected, this command will return an ERR_MULTIPLE_CLIENTS_CONNECTED error. Any requests pending by the user will be closed and the API will close the last connection prior to returning to the user. The user must reconnect after calling this API function by issuing an *RP1210_ClientConnect* call. This support is required for all implementations of the RP 1210 API.

If the hardware device supports a “reset”, then the hardware device will be reset. Either way, the affect of this command is the same as RP1210_ClientDisconnect and the RP1210 variables return to their default value.

Parameter	Description
NCommandNumber	RP1210_ Reset_Device
NClientID	The client identifier for the client that wants to issue the command.
FpchClientCommand	Empty buffer, ignored.
NMessageSize	0

10.11.4.1 *RP1210_ Reset_Device* Return Value

Mnemonic	Description
`ERR_HARDWARE_NOT_RESPONDING`	The device is unable to perform a reset.
`ERR_MULTIPLE_CLIENTS_CONNECTED`	The API was not able to reset the device because more than one client is connected.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.

10.11.5. *RP1210_Set_All_Filters_States_To_Pass*

By commanding "set all filter states to pass" it is implied that the API will allow all messages meant for the application to pass through (clearing all active filters). This support is required for all implementations of the RP 1210 API.

Parameter	Description
nCommandNumber	RP1210_Set_All_Filter_States_To_Pass
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	Empty buffer, ignored.
nMessageSize	0

10.11.5.1 *RP1210_ Set_All_Filter_States_To_Pass* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.

10.11.6 *RP1210_Set_Message_Filtering_For_J1939*

The API can set up a message filter or enable the lower-level components to set up a message filter, through this particular configuration of the *RP1210_SendCommand* function. This support is required for all implementations of the RP1210 API that provide a J1939 interface. See also the command *RP1210_SendCommand* (*RP1210_Set_J1939_Filter_Type* …) which sets filters to either inclusive or exclusive; also see the section on Inclusive and Exclusive filtering.

The filter takes effect once the DLL's J1939 receive buffer is empty.

Successive calls to the *RP1210_SendCommand* function with this *nCommandNumber*, would augment, rather than replace, the set of currently active J1939 filters. The lower-level layers will discard all incoming messages except those that match the associated filter parameters.

Parameter	Description
nCommandNumber	RP1210_Set_Message_Filtering_For_J1939
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifying the filtering parameters. A J1939 filter parameter frame is shown below.
nMessageSize	The number of bytes in fpchClientCommand. Should be a multiple of seven.

10.11.6.1 J1939 Filter Parameter Frame

Bytes	Length	Item	Description
Byte 1	1 byte	Filter Flag	The filter flag (bit field) indicates which of the possible fields in this frame that should be used for filtering. Filtering is effected by matching the appropriate fields to the corresponding fields in the incoming message. The valid values are shown in the table below and can be logically OR'd together. **Mnemonic** — **Comments** `FILTER_PGN` Use PGN value for filtering `FILTER_SOURCE` Use source address for filtering `FILTER_DESTINATION` Use destination address for filtering `FILTER_PRIORITY` Use priority value for filtering
Bytes 2-4	3 bytes	PGN	The PGN that needs to be filtered. This value is used only if the flag specifies FILTER_PGN option. This value is three bytes in length and is in little endian/Intel format.
Byte 5	1 byte	Priority	The priority of the messages that need to be filtered. This value is only used if the flag specifies the FILTER_PRIORITY option.
Byte 6	1 byte	Source Address	The source address of the messages that need to be filtered. This value is only used if the flag specifies the FILTER_SOURCE option.
Byte 7	1 byte	Destination Address	The destination address of the messages that need to be filtered. This value is only used if the flag specifies the FILTER_DESTINATION option.

10.11.6.2 RP1210_Set_Message_Filtering_For_J1939 Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.7 *RP1210_Set_Message_Filtering_For_CAN*

The API can set up a message filter or enable the lower-level components to set up a message filter, through this particular configuration of the *RP1210_SendCommand* function. This support is required for all implementations of the RP1210 API that provide a CAN interface. See also the command *RP1210_SendCommand* (*RP1210_Set_CAN_Filter_Type* …) which sets filters to either inclusive or exclusive; also see the section on inclusive and exclusive filtering.

The filter takes effect once the receive buffer is empty.

Successive calls to the *RP1210_SendCommand* function with this *nCommandNumber*, would augment, rather than replace, the set of currently active CAN filters. The lower-level layers will discard all incoming messages except those that match the associated filter parameters.

Parameter	Description
nCommandNumber	RP1210_Set_Message_Filtering_For_CAN
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifying the filtering parameters. A CAN filter parameter frame is shown below.
nMessageSize	The number of bytes in fpchClientCommand. Should be a multiple of nine.

10.11.7.1 CAN Filter Parameter Frame

Bytes	Length	Item	Description

Byte 1	1 byte	CAN Type	STANDARD_CAN = 11-bit identifier
			EXTENDED_CAN = 29-bit identifier

Bytes 2-5	4 bytes	MASK	The mask indicates which bits in the header need to be matched. A “1” means that the value of the bit is important; a “0” means that the value is unimportant. This will be in big endian format.
Byte 5	4 byte	Header	The Header indicates what value is required for each bit of interest. To stay generic, four bytes are used for both the mask and header regardless of whether Standard or Extended CAN is used. In the case of Standard CAN, only the low two bytes in both the mask and header would be used. This will be in big endian format.

10.11.7.2 Standard CAN (11-bit Identifier) Example

Header = 0481h
Mask = 0183h
The bit pattern is as follows

Bit #	12 - 32	11	10	9	8	7	6	5	4	3	2	1
Header	N/A	1	0	0	1	0	0	0	0	0	0	1
Mask	N/A	0	0	1	1	0	0	0	0	0	1	1

This means that bits 1, 2, 8, and 9 are the only bits of interest in the header of the incoming message; the other bit values are ignored. Furthermore, the bit values of interest must match the mask exactly, to satisfy the filter condition:

Bit 1 = 1
Bit 2 = 0
Bit 8 = 1
Bit 9 = 0

(Notice that the “1” in bit 11 of Header has no effect)

Now with this setting if we receive 4 messages with message headers 681h, 689h, 609h and 9h respectively, only the first two messages will pass through.

10.11.7.3 *RP1210_Set_Message_Filtering_For_CAN* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.

`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.8 *RP1210_Set_Message_Filtering_For_J1708*

The API can set up a message filter or enable the lower-level components to set up a message filter, through this particular configuration of the *RP1210_SendCommand* function.This support is required for all implementations of the RP1210 API that provide a J1708/J1587 interface. See also the command *RP1210_SendCommand* (RP1210_Set_J1708_Filter_Type …) which sets filters to either inclusive or exclusive; also see the section on inclusive and exclusive filtering.

The filter will be set up by supplying a list of module IDs (MIDs) in the *fpchClientCommand*. The lower-level layers will discard all incoming messages except for the ones that are associated with the MIDs supplied in this list. As a clarification for RP1210B, note that the application may pass in an array of MIDs that will all be added to the list. Successive calls to the *RP1210_SendCommand* function with this *nCommandNumber* , would augment, rather than replace, the set of currently active J1708 filters. The lower-level layers will discard all incoming messages except those that match the associated filter parameters. The filter takes effect once the DLL's J1708 receive buffer is empty.

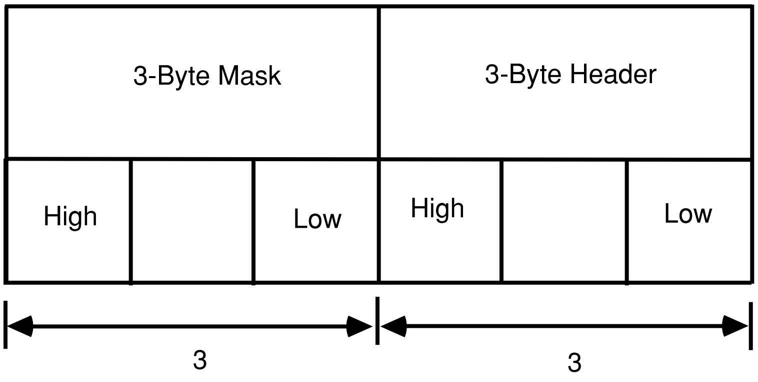
Parameter	**Description**
nCommandNumber	RP1210_Set_Message_Filtering_For_J1708
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifying the MIDs to filter on. An array of MIDs may be provided.
nMessageSize	The number of bytes in fpchClientCommand.

10.11.8.1 *RP1210_Set_Message_Filtering_For_J1708* Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.9 *RP1210_Set_Message_Filtering_For_J1850*

J1850



The API can set up message filters or enable their lower-level components to set up message filters, through the particlar configuration of the *RP1210_SendCommand* function. This support is required for all implementations of the RP1210(B) API that provide J1850 interface.

The filter takes effect once the receive buffer is empty. Successive calls to the *RP1210_SendCommand* function with this *nCommandNumber*, would augment, rather than replace, the set of currently active J1850 filters. The lower-level layers will discard all incoming messages except those that match the associated filter parameters.

Parameter	**Description**
nCommandNumber	*RP1210_Set_Message_Filtering_For_J1850*
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifing the *filtern* parameters.
nMessageSize	Number of bytes in *fpchClientCommand*. It should be a multiple of 6 bytes.

10.11.9.1 RP1210_Set_Message_Filtering_For_J1850 Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.10. *RP1210_Set_Message_Filtering_For_ISO15765*

The API can set up message filter or enable the lower-level components to set up a message filter, through their particlar configuration of the *RP1210_SendCommand* function. This support is required for all implementations of the RP1210(B) API that provide an ISO15765 interface.

The filter takes effect once the receive buffer is empty. Successive calls to the *RP1210_SendCommand* function with this *nCommandNumber*, would augment, rather than replace, the set of currently active ISO15765 filters. The lower-level layers will discard all incoming messages except those that match the associated filter parameters.

Parameter	**Description**
nCommandNumber	RP1210_Set_Message_Filtering_For_ISO15765
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifying the filtering parameters. An ISO15765 filter parameter frame is shown below.
nMessageSize	The number of bytes in fpchClientCommand. Should be a multiple of eleven.

10.11.10.1 ISO15765 Filter Parameter Frame

Bytes	Length	Item	Description

Byte 1	1 byte	CAN / extended address type	STANDARD_CAN = 11-bit identifier
			EXTENDED_CAN = 29-bit identifier
			STANDARD_CAN_ISO15765_EXTENDED = 11-bit identifier with ISO15765 extended address
			EXTENDED_CAN_ISO15765_EXTENDED = 29-bit identifier with ISO15765 extended address
			STANDARD_MIXED_CAN_ISO15765 = 11-bit identifier with mixed addressing

Bytes 2-5	4 bytes	MASK	The mask indicates which bits in the header need to be matched. A "1" means that the value of the bit is important; a "0" means that the value is unimportant. Values are big-endian.
Byte 6	1 byte	Extended address MASK	The extended address mask indicates which bits in the extended address header need to be matched.
Byte 7-10	4 byte	Header	The Header indicates what value is required for each bit of interest. To stay generic here, four bytes are used for both the mask and header regardless of whether Standard or Extended CAN is used. In the case of Standard CAN, only the low two bytes in both the mask and header would be used. Values are big-endian.
Byte 11	1 byte	Extended address header	The extended address header indicates what value is required for each bit of interest. To stay generic one byte is used for both the mask and header, but is only used when the CAN / extended address type is set to STANDARD_CAN_ISO15765_EXTENDED or EXTENDED_CAN_ISO15765_EXTENDED or STANDARD_MIXED_CAN_ISO15765.

10.11.10.2 *RP1210_Set_Message_Filtering_For_ISO15765* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.11 *RP1210_Generic_Driver_Command*

This command allows the client application to send a generic message to its drivers. The API simply passes the message in the *fpchClientCommand* buffer down to the driver(s), if any, associated with the device hardware without intercepting or interpreting it. This support is optional for implementations of the RP 1210 API.

Parameter	Description
nCommandNumber	RP1210_Generic_Driver_Command
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A free form buffer of bytes to be interpreted by the associated driver(s).
nMessageSize	The number of bytes in fpchClientCommand buffer.

10.11.11.1 *RP1210_Generic_Driver_Command* Return Value

Mnemonic	Description
ERR_COMMAND_NOT_SUPPORTED	The command number is not supported by the API DLL.
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialized.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.12 RP1210_Set_J1708_Mode

The API can toggle the mode of the J1708 interface from *Converted Mode* to *Raw Mode*. When a J1708 link is established, the link, by default, is in *Converted Mode*.

Converted Mode is defined as follows: When a J1708 data packet is received; the lower level interface is verifying the checksum byte of the message, stripping off the checksum byte, and passing only validated messages to the client application. When the client application is sending a data packet, the lower level interface calculates the checksum byte and adds the byte to the trailer of the data packet.

Raw Mode is defined as follows: When a J1708 data packet is received, the lower level driver passes the complete message (including checksum), to the application. The checksum byte would neither be inspected nor removed. When the client application is sending a data packet, the application would be responsible for calculating and adding the checksum byte. This mode would expand the diagnostic capabilities of the API by giving a client application the ability to inspect the entire protocol and all J1708 bus traffic.

New to RP 1210B is that multiple clients may be connected in Raw Mode. It was shown that the majority of VDA vendors did not have this limitation anyway.

Whenever this command is called to switch either to Raw or Converted Mode, the send and receive buffers allocated at *RP1210_ClientConnect* are cleared.

This support is required for all implementations of the RP 1210 API that provide a J1708 interface.

Parameter	Description
nCommandNumber	RP1210_Set_J1708_Mode
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0] = CONVERTED_MODE/RAW_MODE
nMessageSize	1

10.11.12.1*RP1210_Set_J1708_Mode* Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.13 *RP1210_Echo_Transmitted_Messages*

The API can toggle the lower level interface to echo transmitted messages back to the client application. See the section on EchoMode. Note that "if and only if, EchoMode is on" will a send message be read back by the same application's call to *RP1210_ReadMessage*. This support is required for all implementations of the RP1210 API.

When Echo Mode is turned on, the send and receive buffers allocated at RP1210_ClientConnect will be cleared.

Turning Echo on introduces a new byte into each incoming message that will indicate to the client whether the incoming message was the result of an echo or was an incoming message. This byte is only available when echo is turned on so the client must remember that all incoming messages will have an extra byte.

Parameter	**Description**
nCommandNumber	RP1210_Echo_Transmitted_Messages
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0] = ECHO_ON/ECHO_OFF
nMessageSize	1

10.11.13.1 *RP1210_Echo_Transmitted_Messages* Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.14 *RP1210_Set_All_Filters_States_to_Discard*

The API can set all message filter states to discard through this configuration of the RP1210_SendCommand function. By "setting all filter states to discard," it is implied that the API will not allow any messages to go through. This support is required for all implementations of the RP1210 API.

Parameter	**Description**
nCommandNumber	RP1210_Set_All_Filters_States_to_Discard
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	Empty buffer, ignored.
nMessageSize	0

10.11.14.1 *RP1210_ Set_All_Filter_States_To_Discard* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.

10.11.15 *RP1210_Set_Message_Receive*

This command is used to enable or disable reception of messages from the specified client. By default a newly created connection has message receive turned on, but the filter state is set to “discard,” so the application will not receive messages.

If the client is already receiving messages turning it on (again) has no effect. The client can stop receiving messages by calling this function and turning reception off. If reception has already stopped then the call will have no effect. When this command is issued and a change of state occurs (off to on, on to off) all messages not already claimed by the application will be lost.

Parameter	Description
nCommandNumber	RP1210_Set_Message_Receive
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0] = RECEIVE_ON/RECEIVE_OFF
nMessageSize	1

10.11.15.1 *RP1210_ Set_Message_Receive* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.

10.11.16 *RP1210_Protect_J1939_Address*

The API has the ability to claim and protect a J1939 address on the vehicle network. When this command is issued, the API will attempt to claim the address that the application has requested. If the address is successfully claimed, the API will continue to protect the address. The API must support the claiming and protection of one address per J1939 client attached to the API.

If the API is forced to concede the address to another node on the network, any blocked calls to *RP1210_SendMessage ()* will be released with an error. If in polling mode only, then the next call to the *RP1210_SendMessage* () will return an error indicating that the address has been lost. The API will also cease to handshake for RTS/CTS transport sessions to the lost address.

After the application has been notified of the lost address, the application can either attempt to claim another address, or invoke *RP1210_Release_J1939_Address* to release the previously claimed address and turn off network management. The application must perform one of these tasks before the API will allow it to send or receive another multi-frame message.

Once an address has been successfully claimed, the API may begin to handshake for J1939 RTS/CTS sessions to the protected address, depending on the parameters that were passed to the *RP1210_ClientConnect ()* function. The API shall not handshake for any messages that are not destined to an address that it is protecting. Upon the loss of a protected address, it is the duty of the API to abort current J1939 transport sessions, and perform necessary cleanup ***prior*** to the notification of the client application. It is not required that the application send this command. The default setting when a J1939 client is connected is for Network Management to be off. If this command is never called, the application will be able to send and receive single frame messages to/from any address on the J1939 vehicle network.

Even if the client application is protecting an address, the API will allow the application to send from any address, determined by the identifier passed to the *RP1210_SendMessage ()* function. An address can also be released by claiming a second address.

Parameter	Description
nCommandNumber	RP1210_Protect_J1939_Address/19d/13h
nClientID	The client identifier for the client that wants to claim a J1939 address.
fpchClientCommand	A pointer to the buffer (allocated by the client application), containing a J1939 Address Claim Frame as described below.
nMessageSize	10

10.11.16.1 J1939 Address Claim Frame

Bytes	Length	Item	Description
Byte 1	1 byte	Address to Claim	The 8-bit address that the client wishes to attempt to claim on the J1939 bus. This address is not limited to the service tool addresses.
Bytes 2-9	8 bytes	Network Mgt Name	The next 8 bytes contain the 8-byte name of the client on the network. This name must be in compliance with the J1939 network management standard and is the responsibility of the client application to assemble.
Byte 10	1 byte	Status Byte	This byte contains the requested status byte for the protect address command. The status request byte is set by the application and can set to the following values: BLOCK_UNTIL_DONE 0 RETURN_BEFORE_COMPLETION 2

10.11.16.2 *RP1210_ Protect_J1939_Address* Return Value

NOTE: If the Status Byte is set to `BLOCK_UNTIL_DONE`, this command will not return until the driver has completed the J1939 address claim procedure.

Mnemonic	Description
`ERR_HARDWARE_NOT_RESPONDING`	The device is unavailable.
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_ADDRESS_CLAIM_FAILED`	The API was not able to claim the requested address.
`ERR_ADDRESS_LOST`	The API was forced to concede the address to another node on the network.

`ERR_BUS_OFF`	The API was not able to transmit a CAN packet due to the CAN hardware being in a BUS_OFF condition.
`ERR_COULD_NOT_TX_ADDRESS_CLAIMED`	The API was not able to request an address.
`ERR_COMMAND_NOT_SUPPORTED`	The command number is not supported by the API DLL.

10.11.17 *RP1210_Release_J1939_Address*

The API has the ability to release a previously claimed J1939 address. When this command is issued, the API will release the claimed address and turn network management functions off. The API will also cease to handshake for RTS/CTS transport sessions and will abort any active RTS/CTS sessions. This command can also be used after the application has been notified of the lost address, to release the previously claimed address and turn off network management.

When a protected address is released, it is the duty of the API to abort current outgoing or incoming J1939 transport sessions. If an outgoing J1939 transport session is aborted by this command, then *RP1210_SendMessage ()* will return an error code. It is not required that the application send this command. The default setting when a J1939 client is connected is for Network Management to be off. If this command is never called, the application will be able to send and receive single frame messages to/from any address on the J1939 vehicle network.

An address can be also be released by claiming a second address via *RP1210_Protect_J1939_Address*.

Parameter	Description
nCommandNumber	RP1210_Release_J1939_Address
nClientID	The client identifier for the client that wants to release a J1939 address.
fpchClientCommand	A pointer to the buffer (allocated by the client application), containing a J1939 Address as described below.
nMessageSize	1

10.11.17.1 J1939 Address Claim Frame

Bytes	Length	Item	Description
Byte 1	1 byte	Address to Release	The 8-bit address that the client wishes to release on the J1939 bus. This address is limited to a previously claimed address.

10.11.17.2 *RP1210_ Release_J1939_Address* Return Value

Mnemonic	Description
`ERR_HARDWARE_NOT_RESPONDING`	The device is unavailable.
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_ADDRESS_RELEASE_FAILED`	The API was not able to release the requested address.
`ERR_BUS_OFF`	The API was not able to transmit a CAN packet due to the CAN hardware being in a BUS_OFF condition.
`ERR_COMMAND_NOT_SUPPORTED`	The command number is not supported by the API DLL.

10.11.18 *RP1210_Set_Broadcast_For_J1708/CAN/J1939/J1850/ISO15765*

TMC has identified a need to have the API maintain a "broadcast" list of messages. These messages would be broadcasted periodically based on the applications' needs. Normally, the user application maintains a broadcast list, however having one in the API makes life for the application much easier by not having to set timers to ensure that a "keep-alive" or recurring message is being transmitted on-time.

The broadcast list will be treated like an abstracted programming array. There is a maximum of 16 entries with entry indexes running from 0 to 15, and there are operations to:

- View an entry in the list of messages.
- Destroy the list entirely.
- Add to the list (append).
- Remove an entry from the list.

Each entry will be made up of:

- A message in the exact format that would be sent to *RP1210_SendMessage*.
- A timer entry in milliseconds for when the message needs to be broadcasted periodically.

NOTE: In J1708 and PLC, RAW_MODE or CONVERTED_MODE play a part in the ability of the message to be a valid message on the data bus. It is suggested that the user application build the broadcast list for a specific mode, and not to change modes (which may result in an invalid message). The API can not guarantee that the message will be sent repeatedly within the requested time-frame. Therefore, the application should give plenty of leeway in the value of the "time" parameter for a broadcast message.

See below for return values for all functions.

Parameter	**Description**
nCommandNumber	RP1210_Set_Broadcast_For_J1708 RP1210_Set_Broadcast_For_CAN RP1210_Set_Broadcast_For_J1939 RP1210_Set_Broadcast_For_J1850 RP1210_Set_Broadcast_For_ISO15765
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A complete broadcast list entry as described below.
nMessageSize	The number of bytes in fpchClientCommand.

10.11.18.1 Broadcast List Entry (*fpchClientCommand*)

Byte	Function	Description
Byte 1	Function to Perform	The function to perform. **Mnemonic** **Function** ADD_LIST Add a message to the list. VIEW_LIST View an entry in the list. DESTROY_LIST Remove all entries in the list. REMOVE_ENTRY Remove a specific entry from list. LIST_LENGTH Returns number of items in list.
Bytes 2..n	Byte 1 = ADD_LIST	fpchClientCommand[2..n] = MESSAGE_TIME_TUPLE record(sent) fpchClientCommand[1] = Entry number where message added (returned). 0xff indicates failure in adding entry to list.

Bytes 2..n	Byte 1 = VIEW_LIST	fpchClientCommand[2] = Number of list entry to view [0..15] (sent)
		fpchClientCommand[1] = Valid List Entry (0x00) or List Entry Does Not Exist (0x01) (returned)
		fpchClientCommand[2..n] = MESSAGE_TIME_TUPLE record (returned)

Bytes 2..n	Byte 1 = DESTROY_LIST	fpchClientCommand[2] = Unused (sent) fpchClientCommand[1] = Number of entries deleted (returned)
Bytes 2..n	Byte 1 = REMOVE_ENTRY	fpchClientCommand[2] = Number of list entry to remove [0..15] (sent) fpchClientCommand[1] = Success (0x00) or Failure (0x01) (returned)
Bytes 2..n	Byte 1 = LIST_LENGTH	fpchClientCommand[2] = Unused (sent) fpchClientCommand[1] = Number of entries in list [0..15] (returned)

10.11.18.2 MESSAGE_TIME_TUPLE Record

Bytes	Function
Bytes 1, 2	Time interval in milliseconds for this message to be broadcast (periodically, see notes). Field is a 16-bit integer in little endian/Intel format [0..65535]. **NOTE:** Anything below 5 milliseconds on a PC can get very difficult to maintain depending on the device's architecture. Also this has the potential to flood the bus at anything under 5 milliseconds for CAN and 10 milliseconds for J1850 or J1587.
Bytes 3, 4	Length of Message (16-bit integer in little endian/Intel format). This would be the value that was given to RP1210_SendMessage in the nMessageSize parameter.
Bytes 5..n	An entire Message as would be presented to RP1210_SendMessage in the fpchClientMessage parameter.

10.11.18.3 List Behaviors

Below are the behaviors of the list.

Mnemonic	**Behaviours of the List**
ADD_LIST	Adds an entry to the end of the list.
VIEW_LIST	Views a specific entry in the list (returns a MESSAGE_TIME_TUPLE).
DESTROY_LIST	Removes all entries from the list.
REMOVE_ENTRY	Removes a specific entry from the list. Trailing list items are moved up in the list.
LIST_LENGTH	Returns the number of items currently in the list.

10.11.18.4 Return Value, All Function Calls

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that something was wrong with the command and/or parameters. Check the value returned in the fpchClientCommand buffer based upon the “Function to Perform” parameter.

NOTE: There is no sanity checking of the values that are passed in, so if you tell J1587 to broadcast 100 times a second a 10-byte message, it’s not going to return an error that the bus’s capacity was exceeded.

10.11.19 *RP1210_Set_BlockTimeout*

New to RP 1210B is the ability to set a value to the amount of time that the user would like a command issued with BLOCKING_IO to timeout. This particular function has already been implemented be several API vendors. The first and second byte of the fpchClientCommand will be multiplied together to get a timeout value (in milliseconds). For example, if the user wanted 200 milliseconds, they would send any combination in the first to bytes to provide a value of 200. To set an infinite timeout, the user would transmit either byte (multiplicand) as a zero. This block should fail if the user disconnects the client.

The command only applies to the current client and applies only to future calls to:

- *RP1210_SendMessage*
- *RP1210_ReadMessage*
- *RP1210_SendCommand*

Parameter	Description
nCommandNumber	RP1210_Set_BlockTimeout
nClientID	The client identifier for the client that wants to issue the command.

fpchClientCommand	fpchClientCommand[0] = 8-bit unsigned value, 0 = infinite block
	fpchClientCommand[1] = 8-bit unsigned value, 0 = infinite block

nMessageSize	2

10.11.19.1 *RP1210_Set_BlockTimeout* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.20 *RP1210_Set_J1708_Baud*

New to RP1210B is the ability to set the J1708 baud rate to a value other than 9600. This assists and speeds the downloading of code, parameters and end-of-line data. This function has already been implemented by several VDA vendors at the request of several OEM’s (with this *nCommandNumber* value). The function was implemented in a way that the API can either change the baud rate immediately (CHANGE_BAUD_NOW), or send a message to the ECU to be communicated with before the change of rate (MSG_FIRST_CHANGE_BAUD).

Note that this function is potentially disruptive to ECU's on the data bus that do not understand anything other than 9600 baud. Use this function only in a controlled environment.

Parameter	Description
nCommandNumber	RP1210_Set_J1708_Baud
nClientID	The client identifier for the client that wants to issue the command.

fpchClientCommand	fpchClientCommand[0] = CHANGE_BAUD_NOW/MSG_FIRST_CHANGE_BAUD
	fpchClientCommand[1] = BAUD_9600/BAUD_19200/BAUD_38400/BAUD_57600

nMessageSize	2

10.11.20.1 *RP1210_Set_J1708_Baud* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialised.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified or command is not supported.
ERR_J1708_BAUD_SET_NONSTANDARD	Another application already has the baud rate set non-standard.

10.11.21 *RP1210_Set_J1939_Baud*

New to RP 1210B is the ability to set the J1939 baud rate to a value other than 250k. This assists and speeds the downloading of code, parameters and end-of-line data. The function is implemented in a way that the API can either change the baud rate immediately, or send a message to the ECU to be communicated with before the change of rate.

Note that this function is potentially disruptive to ECU's on the data bus that do not understand anything other than 250k baud. Use this function only in a controlled environment.

Parameter	Description
nCommandNumber	RP1210_Set_J1939_Baud
nClientID	The client identifier for the client that wants to issue the command.

fpchClientCommand	fpchClientCommand[0] = CHANGE_BAUD_NOW/MSG_FIRST_CHANGE_BAUD
	fpchClientCommand[1] = BAUD_125k/BAUD_250k/BAUD_500k/BAUD_1000k

nMessageSize	2

10.11.21.1 *RP1210_Set_J1939_Baud* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialized.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified or command is not supported.
ERR_J1939_BAUD_SET_NONSTANDARD	Another application already has the baud rate set non-standard.

10.11.22 *RP1210_Set_J1708/J1939/J1850/ISO15765/CAN_Filter_Type*

New to RP1210B is the ability to set filtering to inclusive or exclusive. Currently, RP1210A filtering is only "inclusive". Inclusive filtering forces the application to "ask" the API for each MID, PGN, or CANID that it wants to see. However, with "FilterStates=AllFilterStatesToPass", it forces the application to handle all filtering.

In RP1210B, a new filter mechanism called "exclusive" will be added. This will allow the application to say, "I want to see all messages, except from the engine, etc".

- `RP1210_SendCommand (RP1210_Set_J1708_Filter_Type,     FILTER_INCLUSIVE/FILTER_EXCLUSIVE)`
- `RP1210_SendCommand (RP1210_Set_J1939_Filter_Type,     FILTER_INCLUSIVE/FILTER_EXCLUSIVE)`
- `RP1210_SendCommand (RP1210_Set_CAN_Filter_Type,       FILTER_INCLUSIVE/FILTER_EXCLUSIVE)`
- `RP1210_SendCommand (RP1210_Set_J1850_Filter_Type,     FILTER_INCLUSIVE/FILTER_EXCLUSIVE)`
- `RP1210_SendCommand (RP1210_Set_ISO15765_Filter_Type,  FILTER_INCLUSIVE/FILTER_EXCLUSIVE)`

This variable is per client connection and is per protocol. This means that in a multi-protocol/multi-thread environment, each client would have to issue this command. J1708 filtering also applies if connected to a PLC network. Any filters that have previously been set for the given client connection would be cleared after this call, requiring the client to re-establish new filters as needed.

See also the *RP1210_SendCommand* () calls for *RP1210_Set_Message_Filtering_For_J1939*, *RP1210_Set_Message_Filtering_For_J1708*, and *RP1210_Set_Message_Filtering_For_CAN* and *RP1210_Set_Message_Filtering_For_J1850*, *RP1210_Set_Message_Filtering_For_ISO15765*.

Parameter	Description	
nCommandNumber	RP1210_Set_J1708_Filter_Type RP1210_Set_CAN_Filter_Type RP1210_Set_ISO15765_Filter_Type	RP1210_Set_J1939_Filter_Type RP1210_Set_J1850_Filter_Type
nClientID	The client identifier for the client that wants to issue the command.	
fpchClientCommand	fpchClientCommand[0] = FILTER_INCLUSIVE/FILTER_EXCLUSIVE	
nMessageSize	1	

10.11.22.1 *RP1210_Set_J1708/J1939/J1850/ISO15765/CAN_Filter_Type* Return Value

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified.

10.11.22.2 Example (RP1210A = Inclusive)

```
fpchClientCommand[0] = 128;

RP1210_SendCommand (RP1210_Set_Message_Filtering_For_J1708, nClientID, fpchClientCommand, 1);
```

This tells the API that it wants to see messages from MID 128 (Engine #1). If the application wanted to see messages from Transmission #1, it would have to issue another call. The same call would need to be issued for Tractor ABS, Trailer 1 ABS, etc.

10.11.22.3 Example (RP1210B = Exclusive)

```
fpchClientCommand[0] = FILTER_EXCLUSIVE;
RP1210_SendCommand (RP1210_Set_J1708_Filter_Type, nClientID, fpchClientCommand, 1);
fpchClientCommand[0] = 128;
RP1210_SendCommand (RP1210_Set_Message_Filtering_For_J1708, nClientID, fpchClientCommand, 1);
```

This tells the API that it wants to see everything except from MID 128 (Engine #1).

10.11.23 *RP1210_Set_J1939_Interpacket_Time*

New to RP 1210B is the ability to set the inter-packet time on J1939 Connection Management (RTS/CTS and BAM) messages. Currently, the inter-packet times are standard J1939; however some applications (mostly engineering applications) would like to speed up the inter-packet timing parameters for downloading of new code, configurations, etc.

Parameter	Description
nCommandNumber	RP1210_Set_J1939_Interpacket_Time
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0,1,2,3] = Value in milliseconds for the CM facilities (RTS/CTS;BAM) to set the inter-packet timing. This value is four bytes in length and is in little endian/Intel format.
nMessageSize	3

10.11.23.1 *RP1210_Set_J1939_Interpacket_Time* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialised.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.

10.11.24 *RP1210_Set_MaxErrorMsgSize*

New to RP 1210B is the ability to set the maximum return buffer size for a call to *RP1210_GetErrorMsg* and *RP1210_GetLastErrorMsg*. In RP 1210A, the default and maximum size was 80. This command will allow the API to return a longer and possibly better explanation as to what happened and what the error code meant. The default will still be 80; however the application can tell the API that it is capable of receiving messages that are longer. Note that the application can not ask for values lower than 80.

Parameter	Description
nCommandNumber	RP1210_SetMaxErrorMsgSize
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0,1] = Value in bytes that the application has room for in a call to RP1210_GetErrorMsg (81-65535). This value is in little endian format.
nMessageSize	2

10.11.24.1 *RP1210_Set_MaxErrorMsgSize* Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.

10.11.25 *RP1210_Disallow_Further_Connections*

New to RP 1210B is the ability to disallow another application from issuing a client connection through the vehicle adapter. This allows time-sensitive applications or applications that "must" be the only application with a connection to the data bus to operate unimpeded by other applications. This command can only be issued if there are no other applications using the specified data bus. Once this client connection terminates, other clients will be allowed to connect. This command would be common in end-of-line programming stations. Note that any further client connect calls would come back with the new RP 1210B error code:

```
ERR_MULTIPLE_CONNECTIONS_NOT_ALLOWED_NOW     (455)
```

Parameter	Description
nCommandNumber	RP1210_Disallow_Further_Connections
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	Empty buffer, ignored.
nMessageSize	0

10.11.25.1 *RP1210_Disallow_Further_Connections* Return Value

Mnemonic	**Description**
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialized.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.
`ERR_MULTIPLE_CLIENTS_CONNECTED`	The API was not able to "disallow other connections" because another application is using the VDA.

10.11.26 *RP1210_Set_ISO15765_Flow_Control*

Before long (segmented) messages can be communicated using the ISO15765-2 protocol, this command must be called to indicate the parameters used during flow control communication to one or more devices. For each device, the parameters indicate an association between an incoming (device) CAN-ID (with optional extended address) and an outgoing (tester) CAN-ID (with optional extended address), and the values that are transmitted for BS (block size) and STmin (separation time min.).

When receiving long messages, the outgoing CAN-ID (and optional extended address) is used for the FC (flow control) message that is sent when the interface is receiving a segmented message from the incoming CAN-ID. When transmitting long messages, FC messages received on the incoming CAN-ID (and optional extended address) are matched to the transmission via the outgoing CAN-ID.

For messages to be communicated, the application must also configure the filtering such that messages from the incoming CAN-ID will be accepted.

Single Frame (short) messages – such as broadcast messages - can be sent to or received from a device without recourse to the RP1210_Set_ISO15765_Flow_Control command. Successive calls to this *RP1210_SendCommand* function would augment, rather than replace the list of flow control associations.

Parameter	Description
nCommandNumber	RP1210_Set_ISO15765_Flow_Control
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	A byte array specifying the flow control response parameters.
nMessageSize	The number of bytes in fpchClientCommand. Should be a multiple of thirteen.

10.11.26.1 ISO15765 Flow Control Parameter Frame

Bytes	Length	Item	Description

Byte 1	1 byte	CAN / extended address Type	STANDARD_CAN = 11-bit identifier
			EXTENDED_CAN = 29-bit identifier
			STANDARD_MIXED_CAN_ISO15765 = 11-bit identifier with mixed addressing
			STANDARD_CAN_ISO15765_EXTENDED = 11-bit identifier with ISO15765 extended address
			EXTENDED_CAN_ISO15765_EXTENDED = 29-bit identifier with ISO15765 extended address

Bytes 2-5	4 bytes	Incoming (device) CANID	The Incoming CAN ID indicates the CAN-ID that a flow control response is triggered by (during long message receive) or received on (during long message transmit). To stay generic here, four bytes are used for both incoming and outgoing ID regardless of whether Standard or Extended CAN is used. In the case of Standard CAN, only the low (last) two bytes in both incoming and outgoing ID would be used.
Byte 6	1 byte	Incoming (device) extended address	If the CAN / extended address type is set to STANDARD_CAN_ISO15765_EXTENDED or EXTENDED_CAN_ISO15765_EXTENDED or STANDARD_MIXED_CAN_ISO15765, this byte specifies the extended address byte or ISO15765 network address extension that is associated with the incoming CAN ID. Otherwise, this byte is ignored.
Byte 7-10	4 bytes	Outgoing (tester) CANID	The Outgoing CAN ID indicates the CAN ID (and, optionally, extended address byte) used in the outgoing flow control message (during long message receive) or outgoing packets (during long message transmit).

Byte 11	1 byte	Outgoing (tester) extended address	If the CAN / extended address type is set to STANDARD_CAN_ISO15765_EXTENDED or EXTENDED_CAN_ISO15765_EXTENDED, this byte specifies the extended address byte or ISO15765 network address extension that is associated with the outgoing CAN ID. Otherwise, this byte is zero.
Byte 12	1 byte	BS	The block size reported to the ECU in the outgoing flow control message when receiving segmented transfers. (see ISO15765-2)
Byte 13	1 byte	STmin	The separation time reported to the ECU in the outgoing flow control message when receiving segmented transfers. (see ISO15765-2)

10.11.26.2 *RP1210_Set_ISO15765_Flow_Control* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialized.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.

10.11.27 *RP1210_Clear_ISO15765_Flow_Control*

This command causes the list of flow control associations previously defined by calls to the *RP1210_Set_ISO15765_Flow_Control* command to be cleared.

If this command is executed whilst a long (segmented) message is being communicated, the command will take effect once that communication is complete.

After this command is executed, long messages cannot be sent to or received from any device, until *RP1210_Set_ISO15765_Flow_Control* is called again.

Parameter	Description
nCommandNumber	RP1210_Clear_ISO15765_Flow_Control
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	Empty buffer, ignored.
nMessageSize	0

10.11.27.1 *RP1210_Clear_ISO15765_Flow_Control* Return Value

Mnemonic	Description
ERR_DLL_NOT_INITIALIZED	The API DLL was not initialised.
ERR_INVALID_CLIENT_ID	The identifier for the client was invalid.
ERR_INVALID_COMMAND	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.

10.11.28 *RP1210_Set_ISO15765_Link_Type*

This command applies only to the ISO15765 protocol, and allows the application to specify the coexistence strategy used to filter J1939 messages from the ISO15765 connection in the case where ISO15765 messages are presented on a J1939 datalink.

The options available are:

- **LINKTYPE_GENERIC_CAN***: Generic CAN bus.* No filtering of J1939 messaging occurs. This is the default setting.
- **LINKTYPE_J1939_ISO15765_2_ANNEX_A***: J1939 and ISO15765 coexistence as defined in ISO15765-2 Annex A.* Messages that are not on PGNs 52480, 52736, 55808 or 56404 will be filtered (removed) from the data stream.
- **LINKTYPE_J1939_ISO15765_3***: J1939 and ISO15765 coexistence as defined in J1939/21 and ISO15765-3.* Messages that do not have bit 25 and bit 24 of the CAN identifier both set will be filtered (removed) from the data stream.

The filtering introduced by this command is in addition to the filtering that can be user defined with the *RP1210_Set_Message_Filtering_For_ISO15765* command.

Parameter	Description
nCommandNumber	RP1210_Set_ISO15765_Link_Type
nClientID	The client identifier for the client that wants to issue the command.
fpchClientCommand	fpchClientCommand[0] = LINKTYPE_GENERIC_CAN / LINKTYPE_J1939_ISO15765_2_ANNEX_A / LINKTYPE_J1939_ISO15765_3
nMessageSize	1

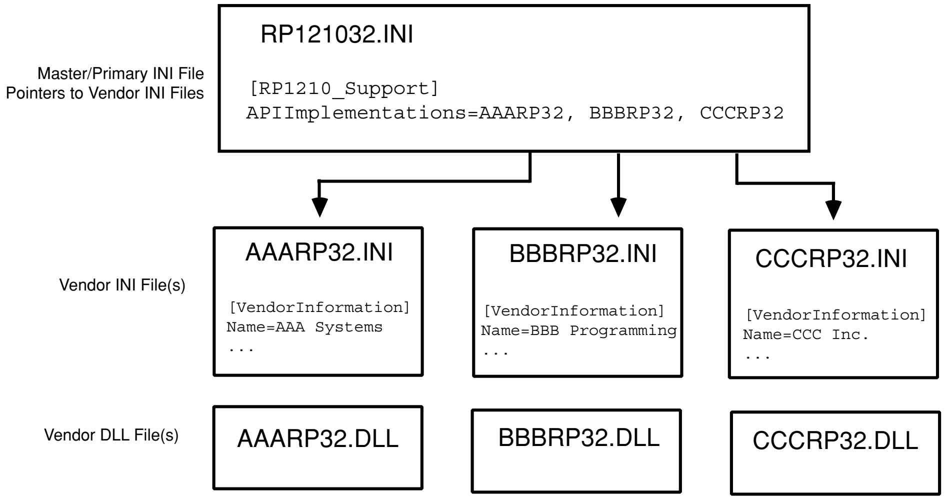
10.11.28.1 *RP1210_Set_ISO15765_Link_Type*

Mnemonic	Description
`ERR_DLL_NOT_INITIALIZED`	The API DLL was not initialised.
`ERR_INVALID_CLIENT_ID`	The identifier for the client was invalid.
`ERR_INVALID_COMMAND`	Indicates that the command number or the parameters associated with it are wrongly specified or the command is not supported.

11.0 INI FILE FORMAT

11.1 FILE FORMAT AND PARSING OF THE RP1210 INI FILES

The following is the structure of how the RP1210 INI system works, and an example of the RP1210 INI files:



Note that there is a "primary" or "master" INI file. This file is called RP121032.INI. The section [*RP1210_Support*] and the INI variable "*APIImplementations*" points to all installed RP1210 API's on the PC. Note that the DLL name is the same as the INI file name (except the extension).

The format of the RP121032.INI file is as follows:

[RP1210Support]

APIImplementations=[string_1],[string_2],...,[string_n]

Each string_i identifies the name of the INI-DLL tuple associated with the vendors' API
An example of the RP121032.INI file would, then, look like:

```
[RP1210Support]
APIImplementations=AAARP32,BBBRP32,CCCRP32
```

```
In this case, the client application has access to the following INI/DLL tuples:
        AAARP32.INI - AAARP32.DLL
        BBBRP32.INI – BBBRP32.DLL
        CCCRP32.INI – CCCRP32.DLL
```

NOTE: In the Windows 3.1 environment, the file RP1210.INI existed with the same format as the RP121032.INI file. This file can safely be ignored except on legacy VDA's supporting Windows 3.1 under RP1210A or RP1210-0.

11.2 VENDOR API INSTALLATION PROGRAM INSTRUCTIONS AND NOTES

11.2.1 Using and/or Creating the RP121032.INI File

The installation program for each vendor-provided INI/DLL tuple would go through the following steps (taking AAARP32 as an example):

Check to see if the RP121032.INI file exists in the directory where the Windows™ operating system is installed. The directory C:\WINDOWS is assumed. The Windows directory can be obtained via the Windows API call *GetWindowsDirectory* (LPSTR lpBuffer, UINT uSize).

If it does not exist, create it ("RP121032.INI"), and write it as:

```
[RP1210Support]
APIImplementations=AAARP32
```

If it does exist:

Retrieve the string for the key APIImplementations using the *GetPrivateProfileString* function. Scan the returned to string check if it contains "AAARP32". If it does not, then append the string ",AAARP32" to the end of the string that was returned and overwrite the APIImplementations key value with the newly appended string. Proceed to the next section on the Vendor INI file.

NOTES:

- It is never acceptable for the installation program to overwrite an existing RP121032.INI file. This has plagued RP1210A application developers and users.
- If you are adding your INI/DLL tuple to the *APIImplementations* variable, it shall be entered at the end of the string. Some INI/DLL tuples have proven only to work at the beginning or at the end of the string. The INI/DLL installation program as well as the PC applications should be intelligent enough to work with the tuple name anywhere in the string.
- Since the RP121032.INI file is one that may, possibly, be written by a number of vendors, it is important that extreme care be taken in the modification of this file by any installation program. An incorrect modification introduced by one installation program could jeopardize the function of the API DLL implementation of another.

11.2.2 Using and/or Creating the Vendor INI File

Each vendor providing an API DLL implementation of the RP1210 API shall also provide an associated INI file. This file would contain information on the hardware devices and protocols supported by the vendor and would be updated every time the vendor's implementation of the RP1210 API is changed to accommodate a new device or protocol. Application programs will parse a vendor-provided INI file for the required connection parameters. This file may also (optionally) contain information to contact the vendor. The file resides in the same directory as the RP121032.INI file. The file format is described below:

11.2.3The [*VendorInformation*] Section of the INI File

The following table describes the [*VendorInformation*] section of the Vendor INI file:

`[VendorInformation]`	Indicates the start of the VendorInformation section.
`Name=[string]`	Where string represents the name of the vendor enclosed.
`Address1=[string]`	Where string represents the first line in vendor’s mailing address—optional.
`Address2=[string]`	Where string represents the second line of the vendor’s address—optional.
`City=[string]`	Where string represents the city that the vendor is located in—optional.
`State=[string]`	Where string represents the state/province of vendor’s location—optional.
`Country=[string]`	Where string represents the country where the vendor is located—optional
`Postal=[string]`	Where string represents the vendor’s postal or ZIP code—optional.
`Telephone=[string]`	Where string encapsulates a telephone number to reach support—optional.
`Fax=[string]`	Where string represents a fax number to reach the vendor—optional.
`VendorURL`	String representing link to vendor web site.
`MessageString=[string]`	Where string is a unique message string for the vendor DLL.
`ErrorString=[string]`	Where string is a unique error string for the vendor DLL.
`TimestampWeight=[number]`	Where [number] represents the weight, per bit, in microseconds of the timestamp.
`AutoDetectCapable=[yes/no]`	If this API is capable of validating that the device(s) an application has requested is physically attached to the computer, the string would be “yes.” Most “smart device” API’s are capable of this feature (if the adapter is physically connected and powered on). Some API’s will have to answer “no.”
`Version=[string]`	Where string reports the API version installed. This is in any format that the VDA vendor wants (i.e., Major.Minor.SubreleaseBuild). This does not mark the version of RP1210 supported. Not being able to “mark” or “version” an INI file has resulted in incompatibility problems.
`RP1210=[string]`	Where string denotes the latest version of RP1210 supported [0,A,B]. If this parameter is present, the developer can assume at least RP1210B is supported.

`DebugLevel=[number]`	This is an optional feature for a DLL provider. If the DLL does not have the capability of initializing in debugging mode, then *DebugLevel*=-1 would be stated. Allows the API to be initialized in debugging/logging mode without having to recompile and relink source code for an application. This is useful when an application seems to be misbehaving. Values to be logged depend on the #: -1 = Debugging is not supported. 0 = No debugging to be accomplished. 1 = Only Connect/Disconnect/Error Messages 2 = Add RP1210_SendCommand calls 3 = Add all Sent Messages (with filtering) 4 = Add all Received Messages (with filtering) See the section on **Debug File Format**.
`DebugFile=[string]`	When *DebugLevel* is greater than zero, the API will write debugging/log information to the file mentioned in this variable. The absolute path (disk/directory) for this file will be included in this variable. The file name should be unique as not to overwrite the log file of anyone else.
`DebugMode=[number]`	When *DebugLevel* is greater than zero, this variable describes how the *DebugFile* is to be written to: 0 = Overwrite (completely destroying previous contents) 1 = Append (write to the end of the file, keeping any previous contents)
`DebugFileSize`	Maximum size in kilobytes that the debug file should be. The default shall be 1024 (1 megabyte) if this variable is not defined.
`NumberOfRTSCTSSessions`	An integer representing the number of concurrent RTS/CTS transport sessions that the API supports per client. This is above and beyond the 1 BAM, 1 RTS/CTS that J1939 calls for. The default will be the J1939 default of 1.
`CANFormatsSupported=[1],[2],[3],[4]`	This adapter allows the user to connect to the CAN data bus using the connection formats [1..4], as defined in *RP1210_ClientConnect*.
`J1939FormatsSupported=[1],[2]`	This adapter allows the user to connect to the J1939 data bus using the connection formats [1..2], as defined in *RP1210_ClientConnect.*
`J1708FormatsSupported=[1],[2]`	This adapter allows the user to connect to the J1708 data bus using the connection formats [1..2], as defined in *RP1210_ClientConnect.*
`ISO15765FormatsSupported=[1],[2]`	This adapter allows the user to connect to the ISO15765 data bus using the connection formats [1..2], as defined in *RP1210_ClientConnect*.
`Devices=[d1],[d2],...,[dn]`	The d's represent the device IDs for the devices supported by this vendor; each d_i is a number between 0 and 255; the device IDs are delimited by commas with no spaces in between.
Protocols=[p_1],[p_2],...,[p_m]	The p_i's represents the identifiers for the protocols supported by this vendor; each protocol identifier is a number between 0 and 255; the protocol identifiers are delimited by commas with no spaces in between; note that, unlike the device IDs, the protocol identifiers have no significance in the application software other than to concoct unique tokens for the different protocols supported by the vendor, and can be arbitrarily assigned.

11.2.4 The [*DeviceInformationdx*] Section of the INI File

For each device listed in the [VendorInformation->Devices] section of the Vendor INI file, the following table describes each of the parameters that will be found in the [*DeviceInformationd$_x$*] section of the Vendor INI file:

[DeviceInformationd$_x$]	Indicates the start of the [*DeviceInformationd$_x$*] section for device d$_x$.
DeviceID=[number]	Where [number] represents a value ranging between 0 and 255. If a device supports multiple interfaces of the same protocol (for example, dual-CAN), each interface (or transceiver) should be assigned a unique *DeviceID* number. The VDA API is responsible for handling each *DeviceID* uniquely and separately.
DeviceDescription=[string,p]	Where [*string*] represents a description of the device. This description is normally the name of the product and is typically listed on the device. When the device can be connected to several ports, the value of [p] is also specified after a comma. If the device does not support several ports, [p] can be left blank. The letter [p] represents the port the device is connected to, the table below lists several values for p. **[p]** **Description** COM1 Computers first serial port COM2 Computers second serial port COMN Computers Nth serial port LPT1 Computers first parallel port LPT2 Computers second parallel port LPTN Computers Nth parallel port PCMCIA PCMCIA Card Device USB USB Capable Device TCP/IP Device Operates Using TCP/IP as a Transport MODEM Device Operates Using a Modem as a Transport WIRELESS Device operates “wirelessly” (802.11/Zigbee/etc.)
DeviceName=[string]	where [*string*] represents the a vendor-specific device name
DeviceParams=[string]	where [*string*] represents vendor-specific parameters, if any—optional

11.2.5 The [*ProtocolInformationp$_x$*] Section of the INI File

For each protocol listed in the [*VendorInformation->Protocols*] section of the Vendor INI file, the following table describes each of the parameters that will be found in that section of the Vendor INI file:

[ProtocolInformationp$_x$]	Indicates the start of the [*ProtocolInformationp$_x$*] section for protocol p$_x$.

ProtocolDescription=[string]
ProtocolSpeed=[string$_1$, string$_n$]
	Where [*string*] represents a description of the protocol
	Where *string*[1] through *string*[n] represents a speed that the protocol can be used at. The following are the strings associated with this entry. Implementation of these speeds are up to the VDA manufacturer and not all adapters will support all speeds.
	[string] **Description**
	all Protocol can be configured for any speed (i.e. CAN registers)
	9600 Common for J1708
	19200 Common for J1708
	38400 Common for J1708
	57600 Common for J1708 10.4Standard J1850 (10.4k Baud)

	41.6 J1850 at 41.6k Baud 125 125k CAN/J1939 250 250k CAN/J1939 500 500k CAN/J1939 1000 1 megabyte CAN/J1939
ProtocolString=[string]	Where [*string*] represents the protocol string expected by the *RP_1210ClientConnect* function
ProtocolParams=[string]	Where [*string*] represents vendor-specific parameters, if any, associated with the protocol—optional
Devices=[k_1,k_2,...,k_t]	Where [k_1,k_2,...,k_t] represents the list of devices associated with this protocol—each k_i is a *DeviceID* and the list is comma-delimited (without any spaces in between).

11.3 STANDARDIZATION OF THE VENDOR INI FILE

In order to provide meaningful descriptions for the users of a vendor specific INI files, this RP defines several *ProtocolDescriptions* and *DeviceNames* that are to be used. In addition to these descriptions, any vendor that implements an RP 1210-compliant API DLL shall obtain a unique vendor name.

The following is a list of protocols and protocol descriptions:

J1708	ProtocolDescription	= J1708 Link Layer Protocol
J1939	ProtocolDescription	= J1939 Link Layer Protocol
CAN	ProtocolDescription	= Generic CAN
PLC	ProtocolDescription	= Power Line Carrier for Trucks (PLC4TRUCKS)
J1850_104k	ProtocolDescription	= 3-Byte Header J1850 at 10.4k Baud
J1850_416k	ProtocolDescription	= 3-Byte Header J1850 at 41.6k Baud
ISO15765	ProtocolDescription	= ISO15765 Link Layer Protocol

11.3.1 Standardized Structure of INI File for Devices with Multiple Ports

Where a device supports multiple interfaces of the same protocol (for example, a device that supports 2 CAN transceivers), each interface (or transceiver) should be assigned a unique *DeviceID* number. This feature has already been implemented by several different VDA vendors in this fashion, however it needed to be documented for future VDA vendor implementations.

NOTE: The VDA vendor is responsible for maintaining all data structures to handle these channels simultaneously. If the VDA vendor does not support simultaneous connections, then they shall note this in their VDA documentation.

API Vendor Header File

The following is the minimal amount of information that should be present in a Vendor API developer header file.

```
//———————————————————————————————
// RP1210A/RP1210B   RP1210_SendCommand Defines
//———————————————————————————————
#define RP1210_Reset_Device                                0
#define RP1210_Set_All_Filters_States_to_Pass              3
#define RP1210_Set_Message_Filtering_For_J1939             4
#define RP1210_Set_Message_Filtering_For_CAN               5
#define RP1210_Set_Message_Filtering_For_J1708             7
#define RP1210_Set_Message_Filtering_For_J1850             8
#define RP1210_Set_Message_Filtering_For_ISO15765          9
#define RP1210_Generic_Driver_Command                      14
#define RP1210_Set_J1708_Mode                              15
#define RP1210_Echo_Transmitted_Messages                   16
#define RP1210_Set_All_Filters_States_to_Discard           17
#define RP1210_Set_Message_Receive                         18
#define RP1210_Protect_J1939_Address                       19
#define RP1210_Set_Broadcast_For_J1708                     20
#define RP1210_Set_Broadcast_For_CAN                       21
#define RP1210_Set_Broadcast_For_J1939                     22
#define RP1210_Set_Broadcast_For_J1850                     23
#define RP1210_Set_J1708_Filter_Type                       24
#define RP1210_Set_J1939_Filter_Type                       25
#define RP1210_Set_CAN_Filter_Type                         26
#define RP1210_Set_J1939_Interpacket_Time                  27
#define RP1210_SetMaxErrorMsgSize                          28
#define RP1210_Disallow_Further_Connections                29
#define RP1210_Set_J1850_Filter_Type                       30
#define RP1210_Release_J1939_Address                       31
#define RP1210_Set_ISO15765_Filter_Type                    32
#define RP1210_Set_Broadcast_For_ISO15765                  33
#define RP1210_Set_ISO15765_Flow_Control                   34
#define RP1210_Clear_ISO15765_Flow_Control                 35
#define RP1210_Set_ISO15765_Link_Type                      36
#define RP1210_Set_J1939_Baud                              37
#define RP1210_Set_BlockTimeout                            215
#define RP1210_Set_J1708_Baud                              305
```

```
//————————————————————————————————
// RP1210B Constants
//————————————————————————————————

#define NULL_WINDOW                         0  // Windows 3.1 is no longer supported.

#define BLOCKING_IO                         1  // For Blocking calls to send/read.
#define NON_BLOCKING_IO                     0  // For Non-Blocking calls to send/read.

#define CONVERTED_MODE                      1  // RP1210Mode=”Converted”
#define RAW_MODE                            0  // RP1210Mode=”Raw”

#define MAX_J1708_MESSAGE_LENGTH            508  // Maximum size a J1708 message can be (+1)
#define MAX_J1939_MESSAGE_LENGTH            1796  // Maximum size a J1939 message can be (+1)
#define MAX_ISO15765_MESSAGE_LENGTH         4108  // Maximum size an ISO15765 message can be (+1)

#define ECHO_OFF                            0x00  // EchoMode
#define ECHO_ON                             0x01  // EchoMode

#define RECEIVE_ON                          0x01  // Set Message Receive
#define RECEIVE_OFF                         0x00  // Set Message Receive

#define ADD_LIST                            0x01  // Add a message to the list.
#define VIEW_LIST                           0x02  // View an entry in the list.
#define DESTROY_LIST                        0x03  // Remove all entries in the list.
#define REMOVE_ENTRY                        0x04  // Remove a specific entry from the list.
#define LIST_LENGTH                         0x05  // Returns number of items in list.

#define FILTER_PGN                   0x00000001  // Setting of J1939 filters
#define FILTER_PRIORITY              0x00000002  // Setting of J1939 filters
#define FILTER_SOURCE                0x00000004  // Setting of J1939 filters
#define FILTER_DESTINATION           0x00000008  // Setting of J1939 filters
#define FILTER_INCLUSIVE                    0x00  // FilterMode
#define FILTER_EXLUSIVE                     0x01  // FilterMode

#define SILENT_J1939_CLAIM                  0x00  // Claim J1939 Address
#define PASS_J1939_CLAIM_MESSAGES           0x01  // Claim J1939 Address

#define CHANGE_BAUD_NOW                     0x00  // Change J1708 Baud
#define MSG_FIRST_CHANGE_BAUD               0x01  // Change J1708 Baud
#define BAUD_9600                           0x00  // Change J1708 Baud
#define BAUD_19200                          0x01  // Change J1708 Baud
#define BAUD_38400                          0x02  // Change J1708 Baud
#define BAUD_57600                          0x03  // Change J1708 Baud
#define BAUD_125k                           0x04  // Change J1939 Baud
#define BAUD_250k                           0x05  // Change J1939 Baud
#define BAUD_500k                           0x06  // Change J1939 Baud
#define BAUD_1000k                          0x07  // Change J1939 Baud
```

```
#define STANDARD_CAN                               0x00  // Filters
#define EXTENDED_CAN                               0x01  // Filters
#define STANDARD_CAN_ISO15765_EXTENDED             0x02  // 11-bit with ISO15765 extended address
#define EXTENDED_CAN_ISO15765_EXTENDED             0x03  // 29-bit with ISO15765 extended address
#define STANDARD_MIXED_CAN_ISO15765                0x04  // 11-bit identifier with mixed address-
ing
#define ISO15765_ACTUAL_MESSAGE                    0x00  // ISO15765 ReadMessage - type of data
#define ISO15765_CONFIRM                           0x01  // ISO15765 ReadMessage - type of data
#define ISO15765_FF_INDICATION                     0x02  // ISO15765 ReadMessage - type of data

#define LINKTYPE_GENERIC_CAN                       0x00  // Set_ISO15765_Link_Type argument
#define LINKTYPE_J1939_ISO15765_2_ANNEX_A          0x01  // Set_ISO15765_Link_Type argument
#define LINKTYPE_J1939_ISO15765_3                  0x02  // Set_ISO15765_Link_Type argument

#ifndef TRUE
#define TRUE                                       0x01
#define FALSE                                      0x00
#endif

//———————————————————————————————————
// RP1210B Return Definitions
//———————————————————————————————————

#define        NO_ERRORS                                        0
#define        ERR_DLL_NOT_INITIALIZED                        128
#define        ERR_INVALID_CLIENT_ID                          129
#define        ERR_CLIENT_ALREADY_CONNECTED                   130
#define        ERR_CLIENT_AREA_FULL                           131
#define        ERR_FREE_MEMORY                                132
#define        ERR_NOT_ENOUGH_MEMORY                          133
#define        ERR_INVALID_DEVICE                             134
#define        ERR_DEVICE_IN_USE                              135
#define        ERR_INVALID_PROTOCOL                           136
#define        ERR_TX_QUEUE_FULL                              137
#define        ERR_TX_QUEUE_CORRUPT                           138
#define        ERR_RX_QUEUE_FULL                              139
#define        ERR_RX_QUEUE_CORRUPT                           140
#define        ERR_MESSAGE_TOO_LONG                           141
#define        ERR_HARDWARE_NOT_RESPONDING                    142
#define        ERR_COMMAND_NOT_SUPPORTED                      143
#define        ERR_INVALID_COMMAND                            144
#define        ERR_TXMESSAGE_STATUS                           145
#define        ERR_ADDRESS_CLAIM_FAILED                       146
#define        ERR_CANNOT_SET_PRIORITY                        147
#define        ERR_CLIENT_DISCONNECTED                        148
#define        ERR_CONNECT_NOT_ALLOWED                        149
```

```
#define      ERR_CHANGE_MODE_FAILED                          150
#define      ERR_BUS_OFF                                     151
#define      ERR_COULD_NOT_TX_ADDRESS_CLAIMED                152
#define      ERR_ADDRESS_LOST                                153
#define      ERR_CODE_NOT_FOUND                              154
#define      ERR_BLOCK_NOT_ALLOWED                           155
#define      ERR_MULTIPLE_CLIENTS_CONNECTED                  156
#define      ERR_ADDRESS_NEVER_CLAIMED                       157
#define      ERR_WINDOW_HANDLE_REQUIRED                      158
#define      ERR_MESSAGE_NOT_SENT                            159
#define      ERR_MAX_NOTIFY_EXCEEDED                         160
#define      ERR_MAX_FILTERS_EXCEEDED                        161
#define      ERR_HARDWARE_STATUS_CHANGE                      162
#define      ERR_INI_FILE_NOT_IN_WIN_DIR                     202
#define      ERR_INI_SECTION_NOT_FOUND                       204
#define      ERR_INI_KEY_NOT_FOUND                           205
#define      ERR_INVALID_KEY_STRING                          206
#define      ERR_DEVICE_NOT_SUPPORTED                        207
#define      ERR_INVALID_PORT_PARAM                          208
#define      ERR_OS_NOT_SUPPORTED                            220
#define      ERR_COMMAND_QUEUE_IS_FULL                       222
#define      ERR_CANNOT_SET_CAN_BAUDRATE                     224
#define      ERR_CANNOT_CLAIM_BROADCAST_ADDRESS              225
#define      ERR_OUT_OF_ADDRESS_RESOURCES                    226
#define      ERR_ADDRESS_RELEASE_FAILED                      227
#define      ERR_COMM_DEVICE_IN_USE                          230
#define      ERR_DATA_LINK_CONFLICT                          441
#define      ERR_ADAPTER_NOT_RESPONDING                      453
#define      ERR_CAN_BAUD_SET_NONSTANDARD                    454
#define      ERR_MULTIPLE_CONNECTIONS_NOT_ALLOWED_NOW        455
#define      ERR_J1708_BAUD_SET_NONSTANDARD                  456
#define      ERR_J1939_BAUD_SET_NONSTANDARD                  457
```

```
//————————————————————————————————
// TMC RP1210B Defined Function Prototypes
//————————————————————————————————

#define DLLEXPORT __declspec(dllexport)

#ifdef __cplusplus
extern “C” {
#endif
short DLLEXPORT WINAPI RP1210_ClientConnect(
       HWND   hwndClient,
       short  nDeviceId,
       char  *fpchProtocol,
       long   lSendBuffer,
       long   lReceiveBuffer,
       short  nIsAppPacketizingIncomingMsgs );

short DLLEXPORT WINAPI RP1210_ClientDisconnect( short  nClientID );

short DLLEXPORT WINAPI RP1210_SendMessage(
       short  nClientID,
       char  *fpchClientMessage,
       short  nMessageSize,
       short  nNotifyStatusOnTx,
       short  nBlockOnSend );

short DLLEXPORT WINAPI RP1210_ReadMessage(
       short  nClientID,
       char  *fpchAPIMessage,
       short  nBufferSize,
       short  nBlockOnSend );

short DLLEXPORT WINAPI RP1210_SendCommand(
       short  nCommandNumber,
       short  nClientID,
       char  *fpchClientCommand,
       short  nMessageSize );

void  DLLEXPORT WINAPI RP1210_ReadVersion(
       char  *fpchDLLMajorVersion,
       char  *fpchDLLMinorVersion,
       char  *fpchAPIMajorVersion,
       char  *fpchAPIMinorVersion );
short  DLLEXPORT WINAPI RP1210_ReadDetailedVersion(
       short  nClientID,
       char *fpchAPIVersionInfo,
       char *fpchDLLVersionInfo,
       char *fpchFWVersionInfo );

short DLLEXPORT WINAPI RP1210_GetHardwareStatus(
       short  nClientID,
       char  *fpchClientInfo,
       short  nInfoSize,
       short  nBlockOnRequest);

short DLLEXPORT WINAPI RP1210_GetErrorMsg(
       short  err_code,
       char  *fpchMessage );
short DLLEXPORT WINAPI RP1210_GetLastErrorMsg(
       short  err_code,
       int   *SubErrorCode,
       char  *fpchMessage );

#ifdef __cplusplus
}
#endif
```

12. DEBUG FILE FORMAT

If the API supports debugging (not a mandatory feature), the API is completely responsible for the format of the debug file.

Notes Concerning Debug File Format:

- The API vendor should completely document the format of the file so that the end-user can readily assist in debugging of a possible application or API problem.
- The file shall be in standard ASCII format readable by “notepad.exe” or “wordpad.exe” and should contain line breaks (CR/LF) at the end of each entry.
- The information should be delineated such that it can be imported into a spreadsheet or programmatically parsed with relative ease.
- Comments in the INI file will help the user to understand what levels of debugging that the API is capable of.

If an API vendor does not support debugging, then the vendor should initialize the INI file variable “DebugLevel” to -1. If an API vendor does support debugging, then the vendor should initialize the INI file variable “DebugLevel” to 0.

It is assumed that if an API supports debugging that it only checks the variable on the first client connection and does not check the variable with each additional client. Therefore, to initialize debugging:

1. All client applications would need to disconnect.
2. The user changes the DebugLevel variable.
3. The user reinitializes the application to debug.
4. The user creates the anomaly (while the API writes to the debug file).
5. The user then assists the appropriate technical support staff to debug.

13.TERMINOLOGY

The following table describes the terminology used in this specification:

Abbreviation	Description
API	Application Programmer's Interface. A documented methodology for programming application programs such that they operate cooperatively with other programs. Software programmers write applications that conform to one or many API's.
Big Endian or Motorola Format	Also referred to in the document as the "most significant byte first format." A format for representing integral bytes of data, conventionally used in Motorola micro-processors, where the bytes are numbered from left to right. The Intel format, called Little Endian, numbers the bytes from right to left.
Client	A logical connection to the onboard vehicle ECU provided by the vehicle communications adapter API through a call to RP1210_ClientConnect .
DLL	Dynamic Link Library. A DLL can be an implementation of an API. A mechanism to link applications to libraries at run-time. The libraries reside in their own (special) executables and are not copied into an application's executable files as with static linking. These libraries are called dynamic-link libraries (DLL) to emphasize that they are linked to an application when it is loaded and executed, rather than when it is linked.
ECM	Electronic Control Module. Synonymous with ECU.
ECU	Electronic Control Unit. The electronic control that will be communicated with (i.e. engine, brakes, transmission). This term will be used throughout this document.
Event-driven Architecture	An operating system architecture that revolves around a messaging scheme governed by system events (a mouse click, a keyboard press are all examples of events). When an application receives an event directed to it, either from the operating system or another application or driver, it executes the code associated with that event. Windows™ architecture is modeled on this paradigm.
INI	Initialization file. This file contains information about every VDA vendors' API, such as protocols supported, communications ports supported, etc. See the section on INI files for the layout of the INI files used by the API.
Interface box	A term usually used in the industry to refer to hardware components that translate between the RS-485 link interfacing with the vehicle datalink, on one hand, and the RS-232 port of the PC on the other. An interface box may be a hand-held diagnostic tool that supports this translation or specially designed RS-485 to RS-232 translation hardware.

Multitasking operating system

An operating system that allows multiple processes to execute at a given time. Such an operating system divides its time and resources between the different running processes. Typically, time-slicing makes it difficult to do dedicated real-time processing with such systems. Windows™ is an example of such an operating system. Compare with non-multitasking operating systems.

Network Protocol

A documented standard for network communications. Examples of network protocol standards in the automotive industry: SAE J1587, J1708, J1939, J1850, ISO15765; CAN (Controller Area Network)

Non-multitasking operating system

An operating system that allows only one process (or task) to execute at a given time. This allows the majority of system resources to be dedicated to the running process, making it eminently suited for real-time processing. DOS is an example of such an operating system. Compare with multitasking operating systems.

PC interface card

A piece of hardware that interfaces with the PC on one hand and the vehicle datalink on the other. Such cards would, typically, plug into one of the expansion slots on the PC.

PCMCIA card

PCMCIA stands for PC Memory Card Interface Association. The term “memory card” is a misnomer as these cards, about the size of credit cards, now support many other functions like modems, Ethernet interfaces, and, conceivably, a vehicle datalinkinterface.

Real-time Processing/Communication

Computation that requires operations to be performed at a high speed (often of the order of milliseconds) and, at times, in an unconditionally prioritized manner. This kind of processing is typical of PC to vehicle datalink communications.

RP 1202 API

A recommended practice by Technology & Maintenance Council (TMC) for vehicle communication under DOS. An implementation of this API may have the ability to support RP1202 type communications as an option.

RP1208

TMC Recommended Practice for PC Service Tool hardware selection.

RS-232

A term usually used in reference to the communication ports available on most PCs. Formally defined as the “Recommended Standard” 232 in the ANSI (American National Standard Institution) specification as “the interface between data terminal equipment and data circuit-terminating equipment employing serial binary data interchange.”

RS-485

A term used in reference to a communication (COM) port available on some PCs. Formally, “Recommended Standard” 485 in the ANSI specification that defines a transmission scheme. This standard is widely utilized in the trucking industry for interfacing with the vehicle datalink.

TCP/IP

Transport Control Protocol/Internet Protocol. Some vehicle adapters allow connecting to a vehicle using this methodology.

USB

Universal Serial Bus. A high-speed serial bus coming into prevalence in the PC to peripheral intercommunications market.

VDA

Vehicle Datalink Adapter. The physical device, when connected to the vehicle data bus, provides translation between the data bus and that vendors' API.

Windows™

A registered trademark (™) of the Microsoft Corporation. This term will refer to operating systems that will be supported under this RP.

TMC Recommended Practice

Proposed RP 1220 (T) **VMRS 036-001-000; 044-011-000**

GUIDELINES FOR COLLISION WARNING AND ADAPTIVE CRUISE CONTROL SAFETY DEVICES FOR USE ON HEAVY TRUCKS.

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to provide guidelines for the selection and specification of collision warning systems and adaptive cruise control used on Class 7-8 combination vehicles.

BACKGROUND

Commercial vehicles represent approximately four percent of the vehicles on the road, seven percent of miles traveled, 12 percent of all traffic fatalities, and 40 percent of the fatal rear-end collisions. According to 2002 General Estimates System[1] (GES) data, 92,241 heavy trucks were involved in 4,542 fatal crashes. Of the 4,897 people killed, 78 percent were occupants of the other vehicle, nine percent were non-occupants, and 14 percent were truck occupants. There were 48,955 non-fatal crashes with 72,590 injuries, of which 75 percent were occupants of the other vehicle, two percent were non-occupants, and 23 percent were truck occupants.

According to a study commissioned by the U.S. Department of Transportation, the costs of large truck crashes in 1997 exceeded $24 billion. Of that, $8.7 billion in productivity losses are estimated, along with $2.5 billion in resources costs, and quality of life losses valued at $13.1 billion. **Figure 1** shows commercial vehicle crashes segmented by type.

Federal officials report that many tractor-trailer accidents are preventable as shown below:

- Road run-off —27 percent.
- Rear-end collisions—25 percent
- Lane change/merge—15 percent
- Drowsy driver—up to 20 percent.

The U.S. Department of Transportation has a goal of reducing fatal crashes on U.S. highways by 41 percent by 2008. For calendar year 2003, fatal crashes involving commercial vehicles accounted for nearly 5,000 of 40,000 total. As means of achieving that goal, the Federal Motor Carrier Safety Administration (FMCSA)—in cooperation with both the automotive and commercial vehicle industries—has funded research, development and testing of both passive and active safety systems that show promise in reducing fatalities and economic loss.

INTRODUCTION

This RP addresses two related types of systems which have the potential to substantially improve fleet safety performance: collision warning system (CWS) and adaptive cruise control (ACC).

A CWS is an automatic system that warn drivers of potentially dangerous situations, thus allowing the driver to take corrective actions. These systems are part of a family of systems categorized as intelligent transportation systems/collision avoidance systems. In general, these systems either passively or actively help drivers avoid crashes. A warning-type system is a passive system which alerts the driver to take an action to avoid a collision, while a control-type system would actively intervene in the control of the vehicle to help avoid a collision. It is important to note that a collision avoidance system cannot overcome

[1] *The General Estimates System (GES) is directed by the National Center for Statistics and Analysis, which is a component of the National Highway Traffic Safety Administration's (NHTSA) Research and Development Division. Data for GES come from a nationally representative sample of police reported motor vehicle crashes of all types, from minor to fatal. The system began operation in 1988, and was created to identify traffic safety problem areas, provide a basis for regulatory and consumer initiatives, and form the basis for cost and benefit analyses of traffic safety initiatives. The information is used to estimate how many motor vehicle crashes of different kinds take place, and what happens when they occur.*

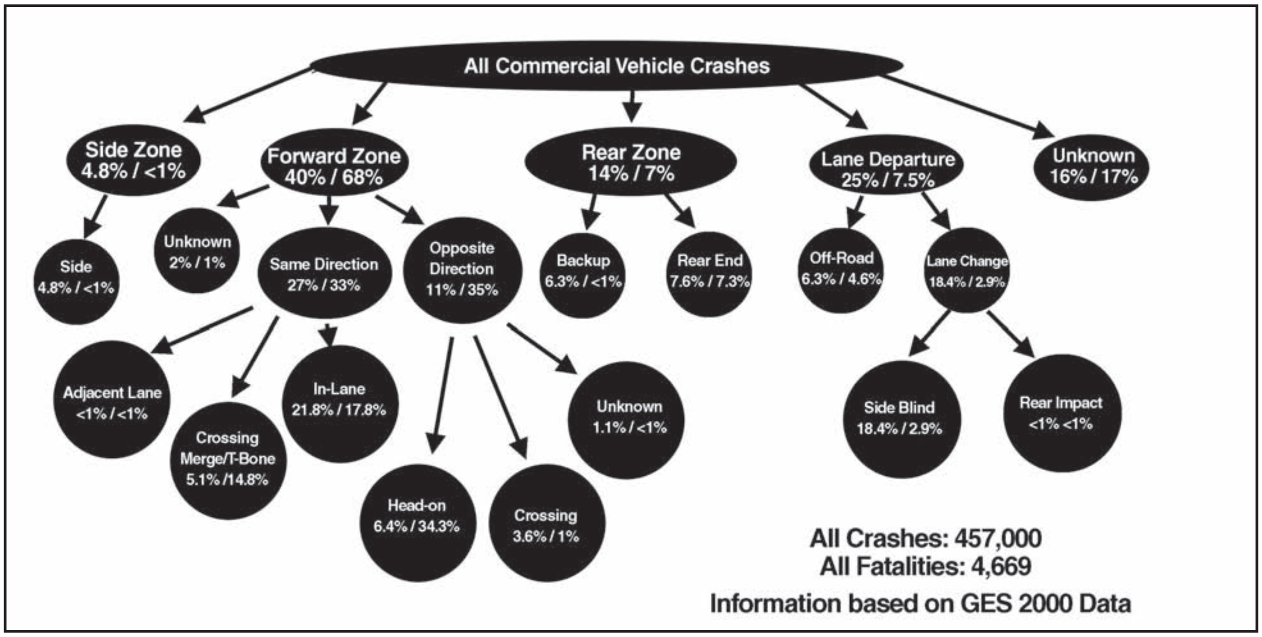


Figure 1: Commercial Vehicle Crashes Segmented by Type of Crash

the laws of physics and, therefore, cannot guarantee that every collision will be avoided.

ACC systems enhance the basic cruise control function. ACC maintains a set speed unless or until a slower vehicle is detected in the lane ahead. In such case, the vehicle automatically slows to match the speed of the slower vehicle and maintain a driver-selected inter-vehicle gap. Both throttle and brakes are automatically controlled to maintain the gap. The degree of braking is limited, however, and the system alerts drivers to intervene with braking in situations requiring rapid deceleration. In this way, users of ACC consider the system to play a key role in increasing driver vigilance and avoiding crashes.

POTENTIAL BENEFITS OF CWS AND ACC

CWS and ACC technology can reduce crashes in vehicles equipped with these systems. In particular, these systems may be able to prevent rear-end and forward impact crashes, reduce the impact speed and, therefore, the crash severity in other instances.

Rear-End Crashes—The CWS can reduce the risk of rear-end crashes by identifying fast-closing situations and providing the driver with additional time to react. ACC maintains a following interval behind the vehicle ahead, thereby providing more time to resolve driving conflicts to reduce the probability of a rear-end collision.

Forward Impact with Objects in Travel Lanes – These crashes occur when a vehicle strikes an non-vehicle object that is in the vehicle's travel lane. Data from the 2003 GES indicates that large trucks were involved in 18,000 crashes of this type in 2003, resulting in 160 fatalities. A CWS can reduce the risk of these crashes by warning the driver of the presence of these objects, thereby allowing additional time to make appropriate avoidance maneuvers.

Other Benefits—Additional benefits of ACC and CWS include:

- reduced number of hard braking events,
- reduced driver workload and fatigue,
- improved reaction time. ACC gives driver additional 0.5 second of perception and 0.5 second reaction time (@60mph = 90 feet). See **Figure 2**.
- reduced insurance rates and deductibles.

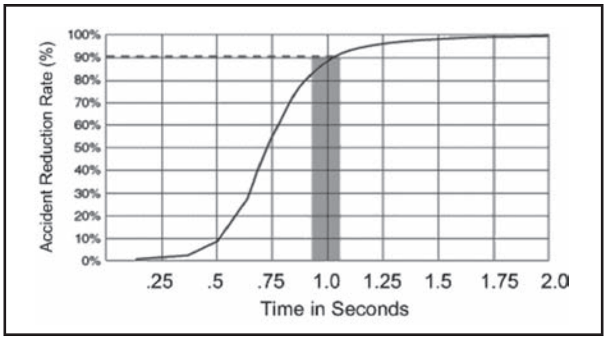


Figure 2

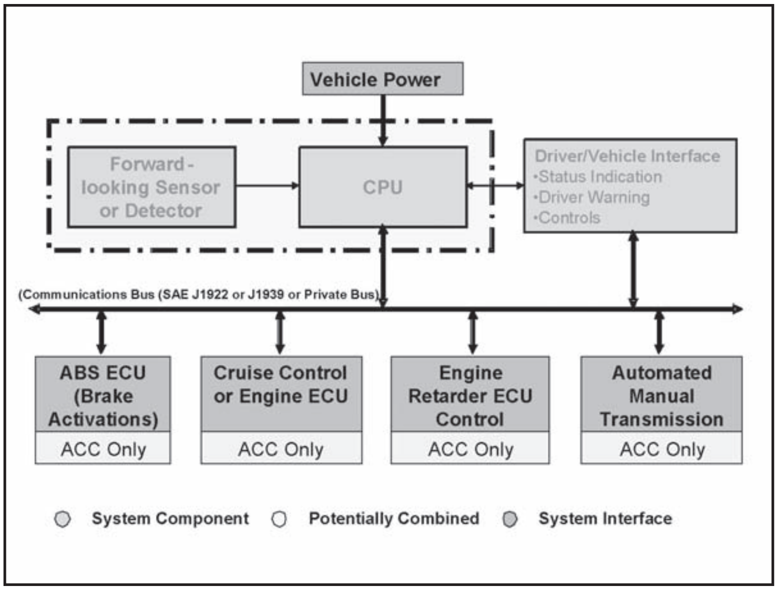

Vehicle Power
Forward-looking Sensor or Detector
CPU
Driver/Vehicle Interface
•Status Indication
•Driver Warning
•Controls
(Communications Bus (SAE J1922 or J1939 or Private Bus)
ABS ECU (Brake Activations)
ACC Only
Cruise Control or Engine ECU
ACC Only
Engine Retarder ECU Control
ACC Only
Automated Manual Transmission
ACC Only
System Component
Potentially Combined
System Interface

Figure 3. Major Functional CWS & ACC Components

- Improved driver comfort, since driver workload is reduced, and increased usage of cruise control is possible across wider speed ranges and in higher traffic densities.

SYSTEM DESCRIPTIONS

The following hardware and software requirements deal directly with detailed hardware functions, environmental and electrical concerns, mounting/installation issues, and software design. **Figure 3** illustrates the major functional components and interfaces of CWS and ACC, respectively.

Collision Warning System

A CWS is an on-board electronic system that monitors the roadway in front of the host vehicle and warns the driver when a potential collision risk exists. Current radar-based systems use algorithms to interpret transmitted and received radar signals to determine distance, azimuth, and relative speed between the host vehicle and the vehicle or object ahead of it in the lane. When the host vehicle is traveling along the roadway, the CWS is active and can warn the driver when a vehicle or object in the host vehicle's lane is within a predefined closing time threshold. If the vehicle or object violates the threshold, the CWS provides a driver warning. Currently, the CWS does not take automatic action to avoid a collision or to control the vehicle; therefore, drivers remain responsible for the safe vehicle operation. As the time interval to the vehicle ahead decreases, the CWS issues a progressively more urgent warning. System manufacturers determine what the warning thresholds will be. **Figure 4** illustrates these progressive thresholds. The CWS also warns the driver of system malfunctions. This system may also interact with ACC to maintain a safe following interval between the host vehicle and the vehicle ahead.

Adaptive Cruise Control

ACC is a “smart” device that maintains a set cruise speed when the road ahead is clear. When the road ahead is not clear due to traffic, ACC automatically slows the host vehicle to match the speed of the closest in-lane slower vehicle ahead (called the target vehicle). ACC also typically provides a headway interval control that lets the driver adjust the gap between the host vehicle and the target vehicle. The ACC may optionally provide an audible warning when it initiates the slowdown command. ACC becomes active when the following two conditions are satisfied:

1. Cruise control has been enabled and a set-speed has been entered by the driver, and;
2. The host vehicle is traveling above a minimum speed threshold (as set by the manufacturer).

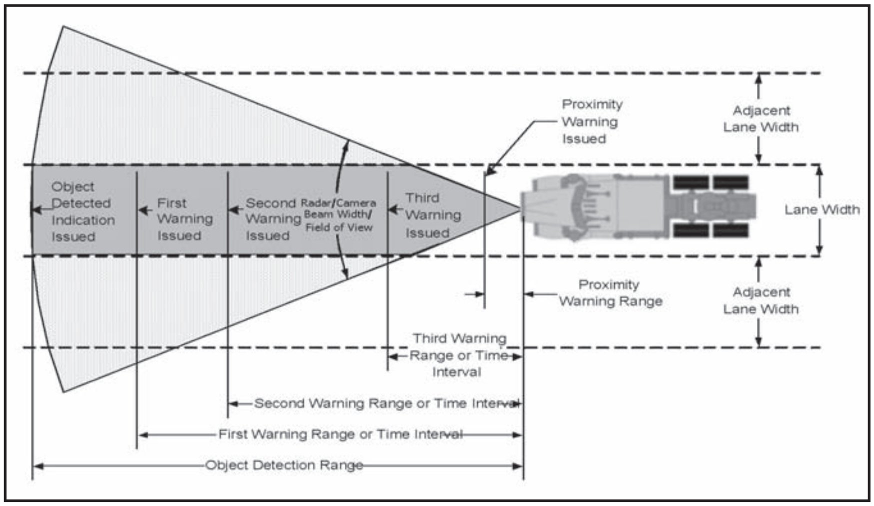

Proximity Warning Issued
Adjacent Lane Width
Object Detected Indication Issued
First Warning Issued
Second Warning Issued
Radar/Camera Beam Width/ Field of View
Third Warning Issued
Lane Width
Proximity Warning Range
Adjacent Lane Width
Third Warning Range or Time Interval
Second Warning Range or Time Interval
First Warning Range or Time Interval
Object Detection Range

Figure 4. CWS Object Detection Ranges & Warning Thresholds

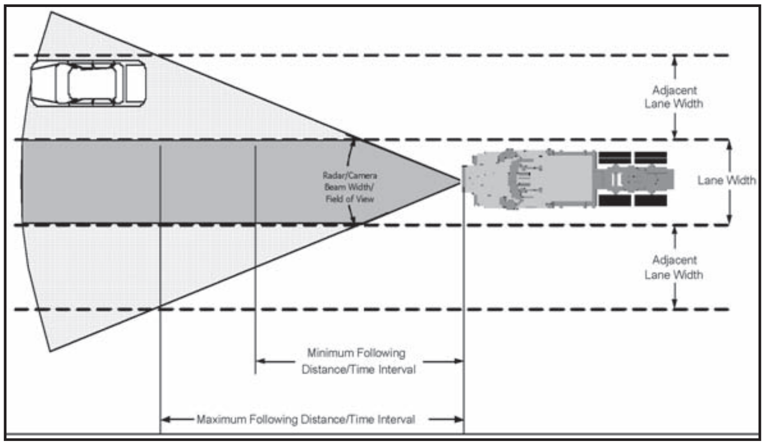

Adjacent Lane Width
Radar/Camera Beam Width/ Field of View
Lane Width
Adjacent Lane Width
Minimum Following Distance/Time Interval
Maximum Following Distance/Time Interval

Figure 5: ACC System Vehicle Following Intervals

Once active, the ACC system maintains the selected headway interval by working in conjunction with the engine management system, cruise control system, and/or the braking system. **Figure 5** illustrates the general thresholds used by an ACC system.

In the absence of a lead vehicle ahead, the system operates just as a conventional cruise control by maintaining the speed set by the driver. The ACC will notify the driver if the system detects a malfunction.

Although ACC systems automatically take action to control the host vehicle's speed, they may not sufficiently decelerate the vehicle in all circumstances to avoid a crash. Depending on the vehicle load, road grade, and vehicle performance parameters, the level of deceleration control typically used by the system is limited to a range between 0.1 and 0.2 g. This is generally sufficient for slowing the host vehicle to match the target vehicle's speed for ACC operation; however, it is not sufficient for panic braking or for bringing the vehicle to a complete stop. As a result, drivers must remain responsible for the safe vehicle operation, and they should apply the service brakes when the needed deceleration exceeds that provided by the engine throttle, engine brake or transmission.

FUNCTIONAL DESCRIPTIONS

Collision Warning System

Normal System Startup Operation—When the driver turns the ignition switch to start the vehicle, the CWS and ACC perform a power-up self-test. The driver scans the warning indicators to determine any system malfunctions. If necessary, the driver may alert fleet maintenance for corrective action. A driver can scan the CWS or ACC system status indicator to verify system operation has commenced.

System Fault Conditions—If a system fault occurs, or the system cannot detect any vehicles or control the host vehicle's speed, the driver is provided with a message via the system status indicator. Some CWS and ACC units may display warning indicators relating to specific faults or other detected conditions.

Progressive CWS Warning/Alert Situations – When traveling at or above the minimum CWS detection speed (on the order of 35 mph) the host vehicle will encounter vehicles and other objects ahead within the travel lane, as is typical in normal driving. As the host vehicle approaches the vehicle ahead, the CWS may provide a series of progressive alerts to the driver. These progressive alerts indicates that the following interval is decreasing and that the driver should take action. These progressive warnings can escalate from audio tone only to audio, visual or other warning modes.

Adaptive Cruise Control

ACC extends the passive nature of the CWS and makes it an active safety and convenience system. By interfacing with the engine throttle and retarder control as well as to the tractor-trailer foundation brakes, the ACC is able to control headway.

ACC provides three major functions:

- Maintain set speed.
- Detect vehicles in the travel lane ahead.
- Match their speed and maintain a predefined headway when slower vehicles ahead are detected.

The primary driver interface to engage and operate the ACC function consists of the standard production cruise controls and a headway selection switch. Using this interface, the driver can do the following:

- Turn the ACC on and off.
- Set the desired cruise speed (set speed).
- Increase set speed by fixed steps.
- Decrease set speed by fixed steps.
- Accelerate to a new set speed.
- Coast (decelerate) to a new set speed.
- Resume a previously set speed.
- Set the desired headway (headway adjustment).

Interval (gap) headway settings offered on current systems are consistent with normal driving practice. Additionally, the accelerator pedal may be used to over-throttle the ACC system. As in standard cruise control, manual braking shall cause the system to go to standby mode. When the ACC is first turned on, the initial headway setting will be set to maximum.

The primary ACC subsystem display may be in a heads-up display or integrated into the instrument panel. The primary ACC display typically includes the following information:

- ACC On/Off
- Set Speed
- Tracking/Not Tracking a Lead Vehicle
- ACC Operational/Failed

The system has a self-diagnosis mode that can detect system faults. If the system is active when a fault is detected, the ACC reverts to manual throttle control. The system does not default to regular cruise control because it could mislead the driver into thinking ACC is still active.

PERFORMANCE EXPECTATIONS

Currently, most CWS units operate at all vehicle speeds. ACC systems only operate above a specified minimum vehicle velocity. This minimum speed generally limits use to those roadways where the speed limit would warrant its use. The application of these systems may be judged by the amount of time a fleet's vehicles operate in these roadway types.

There are currently no performance standards for CWS/ACC on heavy trucks. For the purpose of this RP, the requirements included in the following sections define fundamental CWS and ACC features. The types of requirements listed here are segmented into software requirements, hardware requirements, functional requirements, and operational requirements.

CWS and ACC system manufacturers may include additional functions to augment system capability and features that may be useful beyond minimum system functionality; the operational features that fall into this category are labeled with the term "OPTIONAL." However, in all cases CWS and ACC systems must comply with all existing FMCSA safety regulations. Requirements are designated with an "R;" optional features with a "T."

Software Requirements

Software requirements refer to the embedded software that runs in CWS and ACC systems and controls all system functionality. The CWS and ACC microcontroller or microprocessor continuously runs the system software when the system is active.

T1 The embedded software of CWS and ACC systems can be field-upgradeable via the in-vehicle network connection (i.e., J1587 or J1939) or other common data interface (i.e.., RS-232 or universal serial bus (USB)). Only authorized personnel should upgrade embedded software.

Hardware Requirements

Hardware requirements describe the functionality of the primary physical components of CWS and ACC systems.

R1 **Forward-looking Sensor or Detector—**CWS and ACC systems shall have a sensor or apparatus to detect a vehicle in the frontal aspect of the host vehicle. This sensor or apparatus shall be capable of determining the distance and location of moving vehicles in the host vehicle's travel lane. The CWS and ACC systems should be capable of detecting moving vehicles to at least 100 meters (328 feet) ahead of the host vehicle.

R2 **Driver Vehicle Interface**—CWS and ACC provide a driver vehicle interface (DVI) to permit the driver to interact with the system. A DVI consists of controls and indicators for the CWS and/or ACC systems related to the system operation. The DVI includes progressive audible and visual warnings of a potential collision threat to the host vehicle. CWS and/or ACC systems provide a visual indication of the status of the systems. System status includes operational/non-operational, tracking/not-tracking, and system fault conditions.

T1 **(OPTIONAL) Vehicle Network**—CWS and ACC systems may use the in-vehicle data network (J1708 or J1939) for communication with the vehicle and/or for data communication to data recording or diagnostic devices.

T2 **(OPTIONAL) Stationary Object Detection** —CWS and ACC system may detect stationary objects (vehicles or other obstacles) in the roadway ahead and react to them accordingly. The ability of a sensor or apparatus to detect stationary objects depends on the technology employed (type of radar or laser, or vision system) and the manufacturer's calibration of the system to minimize false alarms (such as differentiating bridge abutments or guardrails alongside the main roadway from stopped vehicles in the lane ahead).

Functional Requirements

Functional requirements described here refer to basic system functionality and operation of the CWS and ACC systems.

R3 CWS and ACC systems shall perform a self-test that checks all major system components and provide system status results to the driver within 30 seconds from Turn_On.

R4 CWS shall detect, track and issue warnings for the following potential pre-collision conditions based on headway interval thresholds when the functional requirements R5 and R6 are met:

- Host vehicle closing on a lead vehicle with constant velocity.
- Host vehicle closing on a lead vehicle with both vehicles decelerating.
- Host vehicle closing on a lead vehicle preceded by a lead-vehicle lane change.
- Host vehicle closing on a stopped lead vehicle.
- Host vehicle closing at a constant speed on a decelerating lead vehicle.

R5 CWS and ACC shall be capable of detecting a vehicle in the host vehicle's travel lane to a distance of at least 100 meters (328 feet) on straight roads and on curves with radius greater than 500 meters (1640 feet).

R6 CWS and ACC systems shall be capable of differentiating between oncoming traffic on two-way curved roadways and same-direction vehicles in the host vehicle's travel lane.

R7 CWS and ACC systems shall function under moderate weather conditions of rain, snow, fog, etc. If the system cannot function in adverse conditions, the driver shall be provided with a clear indication that the system is not functioning.

R8 The CWS and ACC systems shall be capable of differentiating stationary roadside objects such as guardrails, signs, and bridges from moving vehicles in both the host vehicle's travel lane and opposing lanes.

R9 If ACC engages the vehicle's service brakes, the vehicle's brake lights shall be activated in compliance with Federal Motor Vehicle Safety Standard 108.

T3 **(OPTIONAL**) CWS may interact with an ACC to maintain a safe following interval between the host vehicle and vehicle ahead.

T4 **(OPTIONAL**) CWS should use a progressive warning to provide the driver with sufficient time to avoid forward crashes.

T5 **(OPTIONAL**) CWS may issue proximity warnings when the host vehicle is slowly moving forward (less than 2 mph) towards a stationary object or if an object ahead is slowly rolling towards the front of the host vehicle.

Operational Requirements

These requirements define specific ways in which CWS and ACC systems interface with the driver (i.e., Driver-Vehicle Interface (DVI)), and include indicators, displays, and warning methods.

R10 CWS shall utilize different audible tones (e.g., different pitches, patterns, lengths, etc.) and/or visual indicators and/or tactile warnings to provide multiple warnings when an object crosses the warning thresholds.

R11 CWS and ACC systems shall include a visual indicator to show that the system is detecting or tracking vehicles in the lane. Indication may be provided by an instrument panel warning light or an indicator that is integral to each system.

R12 CWS and ACC systems shall use a visual indicator to provide system operational status. The status may be displayed by exception only, i.e. only indicate a fault. This status may be indicated by an instrument panel warning light or an indicator that is integral to each system.

R13 CWS and ACC systems shall use a visual or audible indicator to indicate a system failure or malfunction. This status may be indicated by an instrument panel warning light or an indicator that is integral to the system.

R14 CWS and ACC system indicators should be clearly discernible in direct sunlight and at night.

T6 **(OPTIONAL**) CWS may utilize combinations of audible and visual and tactile indicators to provide multiple warnings of object detection and impending collision.

T7 **(OPTIONAL**) CWS may allow the volume of the audible warnings to be adjusted, but not below a minimum sound level of 65 dBA[2].

T8 **(OPTIONAL**) CWS and ACC systems may provide operational or diagnostic messages, such as "System Operational" on an alphanumeric display to alert the driver of specific conditions or concerns.

[2] For reference: 90 dBA = heavy truck at 10m, 80 dBA = curbside of busy street, 70 dBA = car interior, 60 dBA = normal conversation at 1m (3.28 ft.), and 50 dBA = office noise.

OPERATIONAL ISSUES

A driver may encounter several types of roadways where these systems provide a potential benefit. However, drivers must be aware that these systems do have limitations. Examples are provided below.

Highway Issues

- **Straight and Large Radius Curved Flat Roads**— These are the road types where the CWS and ACC systems could provide the greatest benefit. When the vehicle is operating on these roads, CWS can detect vehicles ahead of the host vehicle and issue warnings to the driver when it detects an excessive closing speed condition. ACC systems will control the host vehicle's speed and interval to the lead vehicle. This control function may reduce driver workload and driver fatigue.
- **Two-way Roads With Sharp Curves**—Current sensors used in CWS and ACC systems have limited ability to see around sharp curves. Sharp curves traversed by the host vehicle may cause lead vehicles to leave the sensor coverage area. When these situations arise, each system has a means of accommodating this loss of target. The CWS systems may not redetect the lead vehicles until the host vehicle enters the curve. The reaction of an ACC system to loss of target is to maintain its current speed until it detects the lead vehicle again. When the target is acquired, the system reverts to the set headway interval.
- **Roads With Steep Hills and Valleys**—Roadway changes in elevation may produce false positive indications with a CWS. When a host vehicle is traveling up a hill, CWS systems may detect objects such as overhead signs. This roadway geometry results in an indication that a stationary object is within the travel lane of the host vehicle. Generally, these systems can differentiate moving and stationary objects such as signs and inhibit the warning. Other roadway features that could cause loss of vehicle track or false alarms include dips in the roadway and hillcrests.

System Adjustments

Safety managers can work with the maintenance department and drivers to explain CWS and ACC system benefits. CWS and ACC system parameters (such as sensitivity and volume) can be configured by fleet personnel for particular vehicle types or drivers.

The CWS and ACC systems may also have the capability to store operational data internally for retrieval by the fleet. This data may be useful in reconstructing an accident or as training feedback in analyzing a driver's performance. Fleet managers may be able to obtain operational data (i.e., number of hard-braking events) from CWS and ACC systems via an in-vehicle network to analyze the data and determine possible systemic problems with their fleet operations (i.e., disproportionate number of close-following events with certain drivers, high number of hard-braking events on certain routes). These in-vehicle networks provide a data "backbone" to the truck, permitting the sharing of various system level data such as engine and transmission performance.

Driver Adjustments

Typically, drivers have some ability to adjust system parameters as well, particularly for volume in audible alerts. See additional driver interface requirements above.

INSTALLATION REQUIREMENTS

CWS and ACC systems have different installation requirements for each vehicle type and model. Depending on the system, CWS and ACC systems may be installed by a truck OEM or as an aftermarket accessory by the fleet or other service personnel. Mounting geometry and installation method are particularly important parts of the system operation as the mounting geometry ensures proper orientation of protective covers and the sensor. The installation process must also ensure the sensor is properly aligned and looking at the road ahead and that the transmitted and/or received signals are not filtered or attenuated by improper materials. Regardless of installation method, CWS and ACC systems should always be installed per manufacturer's recommendations.

OEM Installation Requirements

When installed by a truck OEM, the CWS and ACC component locations and mounting methods are identical for each vehicle model. The OEM provides dedicated wiring harnesses for the connection between the CWS and ACC systems and the vehicle and have ensured that there is sufficient bandwidth on the communication bus to ensure proper error free operation. The man-machine interface is often integrated into the dash and is compatible with the other alarms, controls, and visual indicators in the cockpit. Since audio is often an important part of the warning system employed by each vendor, most OEMs will have ensured that there is sufficient audible separation between different types of alarms and that sound levels are proper for the environment. An additional consideration is the extensive testing and validation that the OEM typically puts any component or system through to ensure that it will function properly and be highly reliable.

Aftermarket Installation Requirements

Considerations of vehicle communications bus load should be considered for those aftermarket installations that interface with the vehicles communication bus—i.e., J1939 or J1922 (and J1708). The installer should check to determine there is sufficient bandwidth to ensure proper operation. The CWS and ACC system's components should be in the proper locations and orientations as described in the installation manual with the key parameters entered into the system and special attention given to the placement, types of materials and thickness of materials, used for sensor protective coverings if used. Aftermarket systems may require periodic maintenance, particularly in the area of component (i.e., radar, camera, antennae) alignment.

Partial OEM Installation Requirements

A number of OEMs offer the option of purchasing only certain components of the ACC / CWS products, such as the communication and power harness. Procurement and installation of the remaining components becomes the responsibility of the fleet and all previous caveats apply. Of particular concern, however, is the alignment. The OEMs have a very precise, automatic method of ensuring that the sensor is aligned with the driving axis, utilizing alignment gear that can cost upwards of $5,000. There are simpler methods, however, of aligning the sensor which are less expensive, but they are not automatic (i.e., a digital level). Manual alignment of the sensor becomes an important and critical procedure and must be done correctly so that the sensor is pointed in the lane ahead.

MAINTENANCE REQUIREMENTS

Preventive maintenance for ACC/CWS systems is quite basic. In terms of failures, users report that the systems are very reliable. For radar-based systems, the only maintenance that should be required on currently available systems is a periodic alignment of the radar sensor to ensure that the radar is pointed along the drive axis of the vehicle. This should be conducted every 50,000 miles and typically takes one hour. Maintenance requirements for vision or IR-based systems are not known at present.

Recommended maintenance tools vary by vendor but can include specialized sensor alignment tools, as well as a digital level. Some alignment processes may require a level surface and for the rear tires to be jacked up.

This preventive maintenance item can be conducted by one person properly trained by the vendor, OEM, or representative in the use of the alignment equipment. It requires a minimal knowledge of truck systems and standard tools. No special training is required other than the vendor's/ OEMs specific instructions on the alignment process.

R15 CWS and ACC shall be serviced periodically in accordance with system maintenance instructions to maintain system functionality. This process maintains the proper alignment of integral CWS and ACC components, such as the radar sensor, camera, and antennae.

T10 **(OPTIONAL)** CWS systems may be transferable from one vehicle to another. ACC systems may be transferable from one ACC-capable vehicle to another ACC-capable vehicle. System recalibration and/or resetting of system parameters should be performed when the system(s) are moved between vehicles. Note: Fleets should be aware that configurations can change from year to year, such that particular systems may not actually be interchangeable between vehicles. Therefore, in moving ACC units from vehicle to vehicle, fleets should consult the ACC provider, engine OEM, and others to ensure compatibility with the various vehicle electronic systems.

TRAINING/DRIVER NOTIFICATION REQUIREMENTS

The driver should be notified that a vehicle is equipped with CWS and/or ACC. Typically this is done with a dash sticker which explains very briefly how each system functions. An example of the wording of the dash sticker is as follows:

- "This vehicle is equipped with a CWS which issues a warning when you are approaching a vehicle ahead too quickly. However, drivers are always responsible for safely braking to slow the vehicle in these situations."
- "This vehicle is equipped with ACC. When activated, the vehicle will detect slower vehicles ahead, match their speed, and maintain a safe gap. Although the system provides moderate braking, drivers are always responsible for braking as needed to ensure safety."

Personal instruction on the systems is also recommended with new drivers, which may include watching a video.

R16 Users shall be provided with a manual and training for CWS and ACC. At a minimum, the user's manual shall include information on the minimum vehicle speed at which the CWS and ACC operate, the conditions under which the systems can/cannot detect and track vehicles, and the types of indicators used to inform the driver if the systems are functioning properly.

T9 **(OPTIONAL)** Video, audio, or computer-based training material may be provided for fleet management and/or drivers.

DATA AND FLEET MONITORING REQUIREMENTS

This section defines the format of data generated by or can be obtained directly in real-time from CWS and ACC systems. Two standards specify in-vehicle data communication in heavy trucks:

- SAE/TMC J1587, "Electronic Data Interchange between Microcomputer Systems in Heavy-Duty Vehicle Applications" (message definition for the J1708 data bus), or
- SAE J1939-71, "Recommended Practice for Control and Communications Network for On-Highway Equipment – Vehicle Application Layer"

T10 **(OPTIONAL)** ACC systems may issue various data messages about the operating states and modes of the system on the in-vehicle data network, which may be custom messages or as defined by J1939 or J1587.

EVALUATION RESULTS

NHTSA has conducted a field operational test for automobiles equipped with CWS and ACC. The results are relevant to commercial vehicles as well as cars. The final report is available at the NHTSA NRD Publications webpage: *http://www-nrd.nhtsa.dot.gov/departments/nrd-12/pubs_rev.html*

ADDITIONAL INFORMATION

Commercial Sources

- Radar—*http://electronics.howstuffworks.com/radar.htm*
- CWS/ACC—*http://auto.howstuffworks.com/cruise-control4.htm; http://a-car.com/adaptive/avoid.html*

Government Sources

The U.S. DOT maintains several websites that contain further information on governmental research, testing and evaluation of CWS and ACC:

- Intelligent Vehicle Initiative: *www.its.dot.gov/ivi/ivi.htm*
- *www.fmcsa.dot.gov/safetyprogs/research/researchpubs.htm*
- *www.fmcsa.dot.gov/facts-research/research-technology/report/report.htm*
- *www.benefitcost.its.dot.gov/its/benecost.nsf/ID1AE47B6EB24A51AA85256E1D0052E890*

Proposed RP 1221 (T) **VMRS 036**

GUIDELINES FOR LANE DEPARTURE WARNING SYSTEMS

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is is to provide guidelines for the selection and specification of a lane departure warning system (LDWS) used on Class 7-8 combination vehicles.

This RP was developed in collaboration with the Federal Motor Carrier Safety Administration (FMCSA) and references related documents developed by the International Standards Organization (ISO).

INTRODUCTION

An LDWS is an onboard system that automatically monitors vehicle position within a travel lane and issues a driver warning whenever a lane crossing occurs without the use of a turn signal. An LDWS is designed to help a driver maintain proper lane position on highway and highway-like roads, and can help prevent crashes involving roadway departure, lane changes/merges and rollover.

Rollover crashes can occur whenever a sudden recovery maneuver is made or when any recovery maneuver is made involving a high-lateral velocity (rate of departure) requiring a relatively large amplitude and/or rapid steering action. The LDWS can help mitigate these situations by alerting the driver to the lane departure before a severe and potential destabilizing recovery maneuver is required.

Each year, approximately 70,000 crashes occur in the U.S. involving single vehicle road departures. Based on U.S. government studies, about half of these events occur when the vehicle simply drifts from its intended travel lane because of driver inattention, drowsiness or other impairment—departures that are potentially preventable using LDWS technology. The other half are typically caused by loss of directional control or are a result of an evasive maneuver, and are, therefore, more appropriately addressed by other onboard safety technologies.

Road departure crashes most often occur during the day (70 percent) under dry and clear weather conditions (75 percent) in a rural area (65 percent). The typical causes of crashes that can be mitigated by LDWS are fatigue and inattention. An LDWS may also:

- Help drivers consistently keep their vehicles within the lane, thereby reducing lane departure crashes.
- Encourage drivers to use turn signals when changing lanes (otherwise, a lane-departure warning sounds).
- Reinforce driver awareness of vehicle lane position to maintain a more central lane position and improve driver attentiveness.

OVERVIEW OF LDWS OPERATION

Fundamentally, an LDWS performs three main functions:

1. Detects lane markings ahead of the vehicle.
2. Monitors vehicle's position within the lane.
3. Warns the driver when the vehicle is diverging (or beginning to diverge) from the lane without turn signal activation.

Today's systems are forward-looking, vision-based devices that use algorithms to interpret video images and estimate vehicle state (lateral position, lateral velocity, heading, etc.) and roadway alignment (lane width, road curvature, etc.). Emerging LDWS technologies may include infraed, electromagnetic, laser, or GPS-based sensors.

Road marking quality varies dramatically from mile to mile, road to road, and region to region. The LDWS is generally quite robust in picking up even poor quality road markings. However, when lane markings are inadequate for detection, the LDWS so notifies the driver.

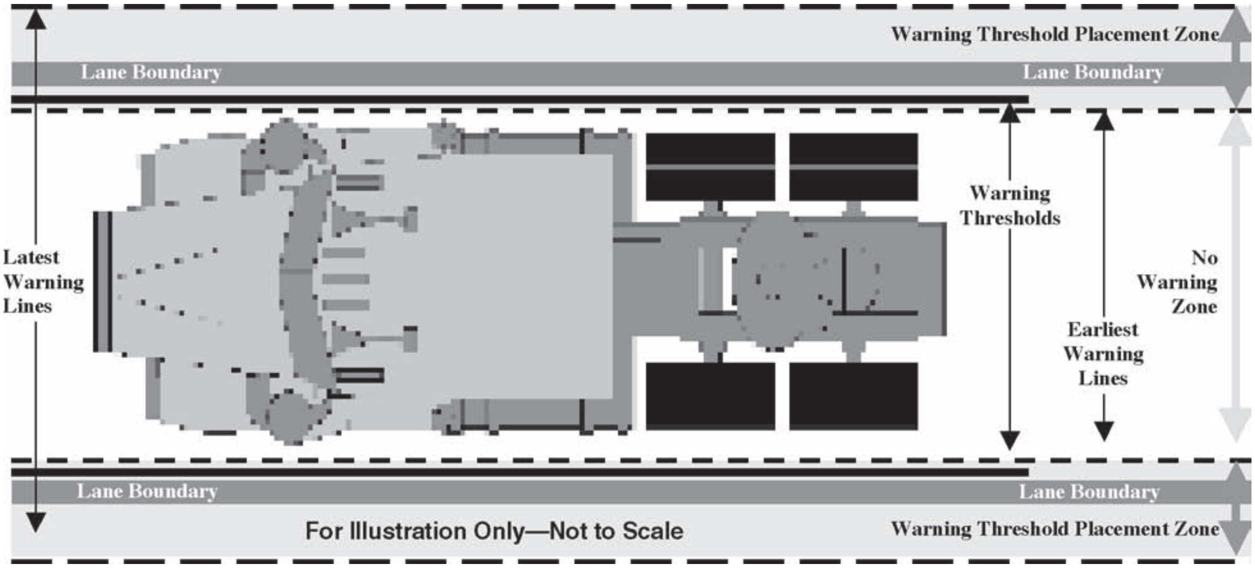


Figure 1. LDWS Warning Thresholds and Warning Threshold Placement Zones

The LDWS do not issue warnings at sections of highways having unusual or irregular lane markings (such as road work sections). Since they are intended for highway use, the systems operate above a threshold speed so as not to create nuisance warnings. An LDWS does not take any automatic action to avoid a lane departure or to control the vehicle. Therefore, drivers remain responsible for the safe operation of their vehicles.

Vision-based LDWS is mounted on the inside windshield within the swept area of the windshield wipers. Wiper use does not degrade LDWS performance.

As shown in **Figure 1**, "earliest warning lines" and "latest warning lines" are defined by the basic system design, and LDWS warnings are based on software-programmable "warning zones." When the vehicle is within the "no warning zone," the system does not issue any position warnings. As the vehicle reaches the warning threshold, the warning is activated. The warning is suppressed if the turn indicator is active, the vehicle is traveling below a threshold speed, or in some instances, when the brake pedal is depressed. The placement of the warning zones may or may not be user adjustable, depending on the product.

The primary physical components of an LDWS are shown in **Figure 2** and described as follows:

- **Lane Boundary Sensor**—A lane boundary sensor detects the vehicle position relative to visible lane boundaries and tracks lane bound-

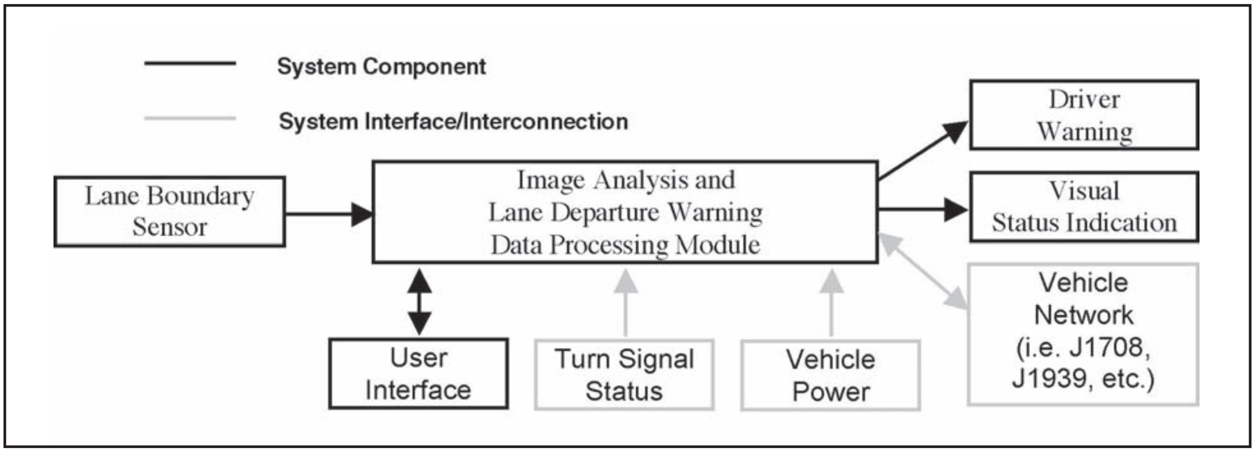

System Component
System Interface/Interconnection
Lane Boundary Sensor
Image Analysis and Lane Departure Warning Data Processing Module
Driver Warning
Visual Status Indication
Vehicle Network (i.e. J1708, J1939, etc.)
User Interface
Turn Signal Status
Vehicle Power

Figure 2: LDWS System Components

aries. The detector is most commonly a vision-based system (camera), and the image of lane boundaries may be black and white or color and may be transferred in either digital or video format to the data processing module.

- **Data Processing Module**—This unit processes data from the lane boundary sensor to establish whether the vehicle is within the no warning zone or has crossed a warning threshold. Using image recognition software and algorithms, the module determines when the vehicle drifts toward an unintended (no active turn signal) lane change and automatically emits a warning for the driver to make a correction.
- **User Interface**—An LDWS typically provides a user interface or driver display unit. Depending on the system design, the user interface consists of various system controls, indicators, and may include an alphanumeric or graphical display.
- **Turn Signal Status**—An LDWS monitors the turn signal status and does not issue a warning if the turn signal direction and the lane departure direction are the same. If they are different, a warning is issued.
- **Vehicle Power**—An LDWS requires power from the vehicle. This is typically 12-volt, unregulated power.
- **Vehicle Network**—Depending on the product, an LDWS may use the in-vehicle data network (J1708 or J1939) to receive vehicle speed information for data communication to data recording or diagnostic devices. This communication may occur in real-time as events are happening, or data may be stored for later communication.

LDWS INSTALLATION REQUIREMENTS

LDWS installation requirements differ depending on the vendor and the type of vehicle on which it is being installed. An LDWS may be installed by a truck manufacturer at the factory, by the dealer prior to delivery or as an aftermarket accessory by the fleet or other service personnel. Trucks may also be prewired at the factory for an LDWS, which can than be installed with relatively little effort after the truck is delivered to the customer.

When installed at the factory, the LDWS lane boundary sensor (i.e. camera) location and mounting bracket attachment method are similar for each vehicle type. The original equipment manufacturer (OEM) provides dedicated wiring harnesses for the connection between the LDWS and the vehicle. The OEM configures and calibrates the system so that it will operate properly upon delivery to the customer.

When installed as an aftermarket accessory, the customer is responsible for physical integration as well as any setup or calibration required. Physical integration consists of:

- mounting the lane boundary sensor (camera),
- mounting the processing unit, and;
- connecting to vehicle wiring.

Camera mounting requirements vary significantly by vendor, so it is important to understand how much time and skill are required, as well as if special tooling is needed. Processing unit packaging and requirements also vary greatly by vendor. They range from units that can be hidden from sight, to DIN-compatible units with a full text/graphical user interface. In every case, the LDWS will need to be connected to vehicle wiring for power and turn signal status. Brake and speed inputs may also be required. It is critical to understand what wiring is required, how it can be accessed (via vehicle bus or discrete wires) and what installation is recommended for a particular make/model vehicle.

Aftermarket installation can take a little as 1/2 hour or as long as several hours, depending on the LDWS. Another important consideration is whether the LDWS can and will be transferred between vehicles—such as might be the case when trucks in the fleet are replaced. Such a transfer is straightforward on trucks equipped with an LDWS amenable to aftermarket installation, but may be more difficult if the LDWS is available only as a factory-installed option.

DAILY OPERATIONAL SCENARIOS

The following describes daily operational scenarios that drivers will encounter when using a LDWS:

- **Normal System Startup Operation**—When the driver turns the ignition switch to start the vehicle, the LDWS performs a power-up self-test, and the driver scans the warning indicator to determine any system malfunctions. If necessary, the driver may alert fleet maintenance for corrective action. When the vehicle reaches the minimum LDWS tracking speed on a roadway with lane boundary markings, lane tracking begins. The driver can scan the LDWS tracking indicator to verify that lane tracking has commenced.
- **Warning/Alert Situations**—When traveling at or above the minimum LDWS tracking

speed and the LDW warning criteria is met, the system issues a warning. This warning may include audible feedback, and/or tactile feedback in the form of a vibrating seat or steering wheel.

- **System Fault Conditions**—When the LDWS cannot track the lane or a system fault occurs, the driver is notified. This condition may be caused by a lack of lane markings, poor quality marking quality, hidden lane markings (dirt, snow), poor visibility (rain, sleet, snow), or a dirty/icy windshield. (Remember, an LDWS cannot issue warnings at sections of highways having temporary or irregular lane markings, such as road work sections.)
- **Calibration**—An LDWS will auto-calibrate, and, depending on the product, notify the driver when calibration is in progress.

LDWS cameras typically view the road through the portion of the windshield swept by the wipers, so that in normal circumstances the driver does not need to take extra effort to keep this portion clean. However, in unusual conditions, system performance may be improved by cleaning the outer glass.

Some systems provide adjustments to tailor operation to fleet policy, driver preferences and current conditions. In these cases, fleet managers can stipulate whether these adjustments are available to the driver or must be fixed at the shop. These adjustments include:

- **Sensitivity Adjustment**—The LDWS may allow earlier or later warnings during the departure sequence.
- **Volume / Intensity Adjustment** — The LDWS may provide an adjustment to allow adjustments to be made to the volume or intensity of the warning signal.

Furthermore, the system may allow the driver to temporarily disable warnings to prevent repeated false alarms—such as might be experienced in construction zones where lane markings may be incorrect. Such manually triggered “off-line” status is typically indicated to the driver through the driver display, and limited in duration to avoid leaving the LDWS in a disabled state.

LDWS MAINTENANCE REQUIREMENTS

Since the LDWS components are typically contained within the truck cab, only minimal maintenance is usually necessary.

It is important to note that a camera-based LDWS needs an unobstructed view of the road ahead. The geometric constraint this places on the camera's location on the windshield is handled at the factory for factory-installed systems, or clearly delineated in the installation instructions for aftermarket systems. The camera is typically placed so as to look through a part of the windshield covered by the windshield wiper swipe pattern, which facilitates maintaining a clear view of the road ahead. The wipers and defrost system are usually all that is required to allow the LDWS to “see” the road ahead. However, persistent windshield spots caused by dirt, bugs or ice may require occasional manual removal to allow for proper LDWS operation. Some systems provide a diagnostic message when such cleaning is necessary.

Perhaps the most significant maintenance issue is windshield replacement. When the truck’s windshield needs to be replaced, camera reinstallation is usually required since it is mounted on the inside of the windshield glass. The time and skill involved in replacing the camera varies depending on the LDWS model selected, and can vary from a few minutes to nearly an hour. The biggest issue with regard to camera reinstallation is the precision with which the camera needs to be placed and pointed. Some LDWS models are self-calibrating, requiring relatively low accuracy in camera placement and pointing, while others require a special mounting jig to guide the camera reinstallation process.

EVALUATION RESULTS

LDWS performance on both trucks and automobiles has been extensively tested in the U.S. and overseas. This section provides an overview of the publicly available results from these tests.

a. **ATA Fatigue Management Technology Field Trial** —Held in partnership with Transport Canada and the American Transportation Research Institute (ATRI), and with support from the DOT Joint Program Office Intelligent Vehicle Initiative, the FMCSA recently completed a fatigue management technology (FMT) pilot tests as part of a study to evaluate combined technologies that could help combat fatigue and the effects of fatigue on truck drivers. Separate tests involving 38 drivers were conducted on trucks in the US and in Canada, to assess the influence different hours-of-service regulations have on driver fatigue and the effective of driver fatigue management technologies. Among the fatigue management technologies tested was a commercially-available LDWS system and an eye blink-based drowsy driver monitor.

Overall, drivers reportedly were very positive about LDWS technology, and said it was very helpful for monitoring both fatigue and distraction. While the experiment was not designed to measure the effects of the individual technologies independently, overall the suite of fatigue management technologies helped improve driver behavior in several important respects. Apparently as a result of the feedback and experience provided by the technology, drivers chose to sleep nearly 1/2-hour longer on average during their days off. When asked the question—"Did the LDWS help you avoid a crash?"—22 of the 38 drivers answered "yes." The report is available from Gerald P. Krueger, Ph.D., CPE, Wexford Group International, 8381 Old Courthouse Road, Suite 300, Vienna, VA 22182-3818. Phone: (703) 749-9134, ext. 204; email gkrueger@thewexfordgroup.com.

b. **LDWS Evaluations Conducted by the Dutch Ministry of Transport—**The Dutch Ministry of Transport sponsored LDWS field trials during 2002 and 2003. The trials focused on professional drivers operating heavy-duty trucks and long-distance buses. The test fleet consisted of 35 trucks and five motorcoaches. Five of the trucks had data recorders to collect detailed information. Several different LDWS products, typical of those offered commercially, were used. Overall, the effects of LDWS technology on traffic safety reportedly were positive. The results indicated that if all trucks in the Netherlands were equipped with an LDWS, approximately 10 percent of injury crashes involving heavy vehicles could be prevented.

Because the Dutch government was investigating using narrower lanes to reduce congestion, LDWSes were evaluated for their ability to help drivers maintain correct lane position in narrow lanes. Truck driving simulator experiments were conducted in which the lane widths on the virtual road varied from 3.5m down to 2.9m. A distraction task was intentionally introduced in order to stress the lane keeping task of the driver. The study results showed that the LDWS system improved lane keeping, particularly on the narrower lane widths. At the same time, however, drivers reported driving required more effort under these stressed conditions.

LDWSes enjoyed a high degree of user acceptance among drivers who used the systems during the on-road evaluation. This was also the opinion of managers at the transport companies involved. About 75 percent of the drivers had positive opinions of an LDWS, and more than 50 percent stated that they would prefer to drive with such a system installed in their vehicle. However, 21 percent of drivers stated that they would prefer vehicles without LDWS.

Drivers noted fewer "startle" reactions during lane departure events by using an LDWS, as they were advised earlier in the event and therefore could respond more gracefully; similarly, reaction times to take corrective action were reduced. About 60 percent of drivers reportedly concluded that the system caused them to pay more attention to the driving task. Increased comfort levels were noted as well.

On the down side, 20 percent of the drivers reportedly said that the system could *cause* a startle response which might be worse than crossing the lane line; this factor was seen as potentially being related to the loudness of the audible warning. Furthermore, drivers perceived 25 percent of the warnings as needless, false alarms. Both warning volume and sensitivity were not adjustable by the drivers.

c. **Other Studies**—In the U.S., LDWS technology was tested on automobiles in NHTSA's Road Departure Crash Warning Project. LDWS technology was also evaluated on trucks in the Mack Trucks Intelligent Vehicle Initiative LDWS Field Test, sponsored by FMCSA. Both projects finished data collection in 2005 and the evaluation of data is underway. Published results are expected in early 2006.

FLEET EXPERIENCE WITH LDWS TECHNOLOGY
Four fleets with LDWS experience shared information with TMC regarding the process of evaluating and integrating an LDWS into fleet operations. Of the four fleets that evaluated an LDWS, three have

adopted a system; the other is still conducting evaluations. Among the four fleets, cost reportedly was not raised as a factor in their purchase decision, which may relate to the relatively low cost of an LDWS as compared to other active safety systems. A summary of their comments follows:

Implementation Case Study—Fleet 1

Fleet 1 is equipping their entire 1000 tractor fleet with an LDWS. As of May 2006, they have over 600 of their 1000 tractors equipped with an LDWS and installations are ongoing for the rest.

Fleet 1 was reportedly driven by a top-down philosophy to create the strongest safety technology strategy possible; therefore, they did not conduct a rigorous quantitative or analytical evaluation prior to purchase. In addition, they expect the LDWS to play favorably in future insurance package negotiations.

Fleet 1 first evaluated LDWS technology with carefully selected drivers. After securing driver acceptance, the fleet implemented an internal marketing plan to bring along all company departments.

In initial tests, Fleet 1 noted significant reductions in relevant crashes. In 2004, they experienced 23 total crashes that could be categorized as "LDWS preventable." As of the end of 2005, Fleet 1 had eight such crashes with none of the involved trucks being equipped with an LDWS. Fleet 1 reports the system is effective with fatigue-related crashes, even though drivers are unlikely to admit to being drowsy. The company also noted that maintenance issues were very minor, and that the LDWS has had a positive impact on driver retention.

Implementation Case Study—Fleet 2

Fleet 2 operates a fleet of cryogenic tankers using 400 tractors. Of the 400, about 180 are equipped with an LDWS as May 2006. Fleet 2 reports it is very safety conscious and are strong believers in LDWS technology and any other crash countermeasures. Fleet 2 conducted surveys of 117 drivers of 117 tractors, who had used the system for between 5-18 months. Based on their experience, Fleet 2 reported:

- Most normally drive with the system enabled.
- Most felt the warnings come at the right time.
- The most typical false alarm rate was 'a few times a week."
- A strong majority felt the system was useful even with false alarms.
- A strong majority felt the system could help prevent crashes.

To make the case for an LDWS, Fleet 2's vice president for operations reportdly presented a strong rationale for safety technology at the Board level within the company to get their philosophical buy-in and proceed in a top-down manner. As of May 2006, Fleet 2 had accumulated approximately 18 million miles with an LDWS, and has experienced no LDWS preventable crashes with tractors so equipped.

Implementation Case Study—Fleet 3

Fleet 3 operates as a nationwide truckload carrier serving all 48 states using 400 tractors. Fleet 3 is reportedly very safety conscious and is a strong believer in LDWS technology and any other crash countermeasures. Fleet 3 currently has installed about 280 LDW systems on units having accumulated about 45 million miles of operation. These units have experienced no LDWS-preventable crashes. Fleet 3 conducted surveys of about 120 drivers who had used the system from between 5-18 months. Based on their experience, Fleet 3 reported:

- Most normally drive with the system enabled.
- Most felt the warnings come at the right time
- The most typical false alarm rate was 'a few times a week."
- A strong majority felt the system was useful even with false alarms.
- Over 70 percent said an LDWS made them better, safer drivers.
- A strong majority (98 percent) felt the system could help prevent crashes.

Implementation Case Study—Fleet 4

As of May 2006, Fleet 4 is evaluating LDWS technology from two different vendors. For each product, two drivers drove with the system for 30 days. Fleet 4 is reportedly experiencing a wide range of driver opinions —from strong positives to strong negatives. One driver who is a trainer for new drivers particularly thought the system would be useful for those less experienced. Fleet 4 noted that an LDWS can help trainers evaluate trainee performance objectively.

Fleet 4 reported the need to address several questions in their evaluation, including:

- For systems equipped with a display, should the display be used as an enhancement, or turned off to minimize potential distractions?
- Should systems be equipped as always on (with ignition), or should drivers have the ability to turn it off?
- How easily can an LDWS be transferred to other tractors when older tractors are being sold?

QUESTIONS FLEET MANAGERS SHOULD ASK LDWS VENDORS

This section offers a list of questions fleet managers should consider asking when evaluating various LDWS options for their particular operation.

a. Installation Questions

- Is the LDWS available as a factory-installed option of the make and model of truck under consideration?
- Can the make and model of truck under consideration be outfitted at the factory with the required wiring harness, so fleet maintenance personnel can install the system after the truck is delivered?
- Can the entire LDWS be purchased directly from the vendor and install by fleet personnel as a retrofit?
- Is it possible to transfer the LDWS to another vehicle?
- How much time and effort is required to retrofit the LDWS on a new truck, or transfer it between trucks?
- For aftermarket installation, what is the installation support provided by the vendor? Is remote or on-site support available?
- For either factory-installed or aftermarket systems, what if any post-delivery calibration of the LDWS is necessary?
- Is the hardware and/or software comprising the LDWS upgradeable in the field, or must the LDWS, or even the entire truck, be returned to the factory for upgrades? Can fleet personnel perform such an upgrade, or is special training required? How likely is an upgrade needed?

b. Maintenance Questions

- What maintenance does the LDWS typically require, and how often is it required?
- What level of skill is required to perform the necessary maintenance?
- Does the LDWS provide diagnostics to indicate when maintenance is required?
- What happens if the windshield is broken and needs replacement? Can the fleet reinstall the camera and perform whatever calibration is necessary or must that be done by a qualified technician either on-site, at the dealer or at the factory?
- What are the terms of the warranty available on the LDWS system? Does the vendor offer a maintenance contract?
- What training material and support is available for fleet maintenance personnel?
- Has the system been designed and tested to meet SAE J1455, “Joint SAE/TME Recommended Environmental Practices for Electronic Equipment Design for Heavy-Duty Trucks?”
- Has the system been designed and tested to meet any OEM-driven trucking temperature, environmental, and EMC/EMI Requirements?

c. Operation Questions

- Will the LDWS operate on the roads and in the conditions the fleet typically operates on?
- What are the limitations of the LDWS detecting lane markings on curved sections of roadway?
- Where/when does the LDWS not operate well?
- How does the LDWS perform in adverse weather conditions? How much influence do fog, rain, snow, wet pavement and low sun angle have on LDWS performance?
- Does the system automatically disable when conditions are inappropriate for warnings? Does it return to an active operating status when the conditions clear up?
- Can the LDWS be turned off, either temporarily or permanently by the driver? If so, can the driver’s ability to turn the system off be limited or prevented? If so, how?
- Is there a way for the driver to prevent the LDWS from providing warnings for a short period of time when warnings are inappropriate – e.g. in a construction zone?
- Are there vehicle velocity limitations within which the system will operate?
- What form of feedback does the LDWS provide to the driver regarding the operating status of the LDWS?
- Does the LDWS provide specific indications of the cause when it isn’t operating, such as a message that the windshield needs cleaning?
- How does the LDWS handle obscured or missing lane markings? What happens when only one lane marker is visible? Does the system continue operating? Does it inform the driver of its limited operating status?
- Does the LDWS suppress warning during intentional lane changes signaled by the turn indicator?
- How is the driver warned of unintended lane departures? Audible warnings? Physical warnings (e.g. vibrating seat)?
- What, if any, graphical displays are provided

by the LDWS?

- Are the volume and/or intensity of these warnings adjustable, either by the driver or fleet management?
- How does the LDWS determine whether to provide a warning or not? Is it based solely on the position of the vehicle (i.e, when the tire touches the lane boundary) or is the vehicle's rate of departure taken into account? In other words, does the system warn earlier during rapid unintended departures than during slow drifts?
- Is the warning threshold (i.e., the magnitude of a departure required to elicit a warning) adjustable either by the driver or fleet-wide?
- If the warning threshold or intensity is adjustable, how are these adjustments made? Through an operator interface in the field, or at the factory? Can drivers be prevented from tampering with these adjustments once they are set?
- Does the LDWS distinguish warnings for crossing a solid vs. a dashed boundary?
- Does the LDWS provide warnings that distinguish between directions of departure (left vs. right)? If so, how are warnings different?
- Does the LDWS provide provision for "curve cutting" to allow for a slight delay in warnings when the driver appears to be intentionally cutting a bit to the inside of the lane on curves?
- What, if any, extra features does the LDWS provide? These might include:
  - Real-time driver alertness feedback based on consistence of vehicle lateral control.
  - Warnings to consider taking a break when alertness appears to drop, or when the vehicle has been operating continuously for an extended period of time.
  - Continuous logging of driver performance.
  - Real-time reporting of driver performance using a wireless onboard communication system.
- What training material is available to help educate drivers on the operation of the LDWS?

d. Fleet Management Questions

- Can fleet policies for sensitivity and volume be set and the driver "locked out" from overriding these settings?
- What type of data does the system store or broadcast?
- Does the LDWS log data about driver and system performance?
- If so, what is logged? Some systems log information like number of lane excursions, time spent intruding into adjacent lanes, and even frequency of unsignalled lane changes - all of which may be helpful for assessing driver safety behavior.
- If the LDWS does log this safety data, are adequate safeguards provided to prevent unauthorized access to it?
- How can the logged data be accessed?
- Can data be accessed remotely while the truck is on the road—i.e., via a wireless communication link—or does reading the data require local access to the truck?

e. Evaluation Questions:

Can I get a demonstration of the LDWS on type of roads my trucks operate prior to committing to purchasing? What are the terms and duration of this demonstration?

Is there a way to evaluate the performance of the LDWS and the performance of my drivers with and without the LDWS?

What if any performance metrics are available to support such evaluation? How are they collected and analyzed?

Can such evaluations be done continuously, or is special equipment required that isn't permanently installed with the LDWS?

Is there a way to provide incentives for my drivers who score well on the performance metrics as a way to reward safe driving and/or penalize potentially dangerous behavior like unsignalled lane changes or inconsistent lateral control?

TMC Recommended Practice

Proposed RP 1222 (T) **VMRS None**

GUIDELINES FOR HEAVY-DUTY VEHICLE STABILITY SYSTEMS

PREFACE

The following Recommended Practice is subject to the Disclaimer at the front of TMC's *Recommended Engineering Practices Manual.* Users are urged to read the Disclaimer before considering adoption of any portion of this Recommended Practice.

PURPOSE AND SCOPE

The purpose of this Recommended Practice (RP) is to provide guidelines for the selection and specification of vehicle stability systems used on Class 7-8 combination vehicles.

INTRODUCTION

Vehicle stability systems have been shown to help mitigate or prevent crashes attributable to rollover and loss of control, according to data derived from the 2002 and 2003 General Estimates System (GES) Accident database.[1] The Federal Motor Carrier Safety Administration (FMCSA) reports that there were 10,200 rollover crashes involving heavy trucks in 2002., resulting in 190 fatalities. FMCSA also reports there were about 9,600 loss-of-control heavy vehicle crashes in 2003, resulting in 230 fatalities.

Many loss-of-control crashes are caused by aggressive driver control action, steering to avoid another vehicle and overcorrection from a lane departure. It has been reported that combination vehicles have the highest crash cost per vehicle over their operational life because of the high number of miles traveled and severity of crashes. (Wang, Knipling, Blincoe, 1999).

There are three main types of stability systems in use today: roll stability advisor (RSA), roll stability (RS), and electronic stability (ES). RSA systems advise drivers when there is an increased risk of vehicle rollover. RS and ES systems intervene by assuming partial control when a vehicle reaches a risk of rolling over or losing lateral stability. ES systems provide RS functions, but the reverse is not true.

ES systems automatically brake individual wheels to prevent the vehicle from either changing too quickly (spinning out or over-steering) or not changing quickly enough (plowing out or understeering). Oversteering may cause a combination vehicle to jackknife. ES systems cannot increase available traction, but they maximize the possibility of maintaining vehicle control during extreme maneuvers by monitoring the driver's natural reaction of steering in the intended direction in oversteer and understeer situations.

Depending on vehicle manufacturer and application, a stability system may offer either the RS feature alone, or the ES feature, which serves to mitigate both yaw and roll stability events. Stability systems cannot prevent all rollover or loss-of-control events, such as tripped rollovers and those caused by sudden turns at high speed and travel on cross-sloped shoulders. These systems can only function within the limits of physics.

System Effectiveness and Operational Vocation

Stability system effectiveness can vary based load, driving conditions, vehicle configuration and drivers.

- **Load**—Operations that carry heavy loads with a high center of gravity (CG) will have a greater tendency for loss-of-control incidents (i.e., tankers, logging, steel roll, cement, etc). Operations that handle dynamic loads that tend to shift during driving or braking, such as liquid or hanging loads, are also at increased risk in certain driving scenarios.
- **Driving Conditions**—Operations that tend to drive more frequently in inclement weather

[1] *The General Estimates System is directed by the National Center for Statistics and Analysis, which is a component of the Research and Development division of the National Highway Traffic Safety Administration (NHTSA). GES data comes from a nationally representative sample of police-reported vehicle crashes of all types, from minor to fatal. GES, which began operation in 1988, identifies traffic safety problem areas, provides a basis for regulatory and consumer initiatives, and forms the basis for cost and benefit analyses of traffic safety initiatives. The information is used to estimate how many motor vehicle crashes of different kinds take place, and what happens when they occur.*

(i.e., snow, ice, rain, etc.) run greater risk of jackknife and loss of control incidents.

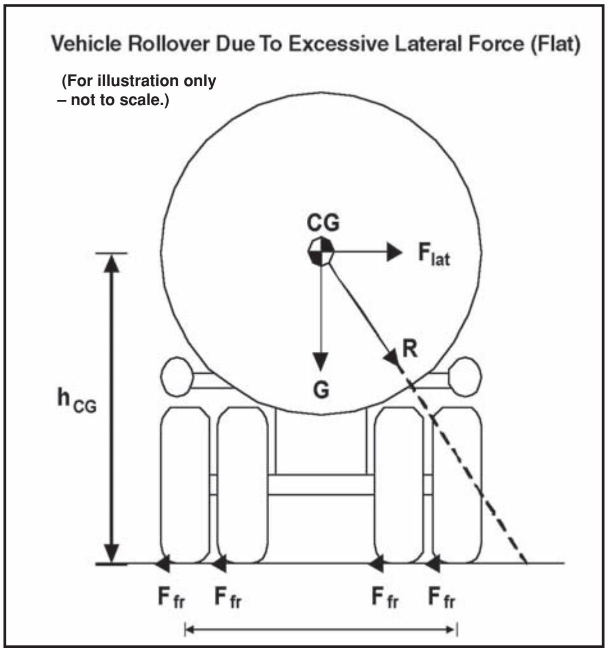
- **Combination Vehicle Configuration**—There are a number of configuration variables that are important including frame and suspension stiffness, track width, and available braking power.
- **Drivers**—Driver experience, route familiarity, attentiveness, and frequency of load/vehicle change are all important factors.

FUNCTIONAL DESCRIPTION OF EVENTS AND CORRESPONDING COUNTERMEASURES

A. Roll Stability Event and Countermeasures

An RS event can occur when a vehicle begins to tip over while changing direction. The lateral (side) acceleration creates a force at the center of gravity, "pushing" the vehicle horizontally. The friction between the tires and the road opposes this force, yet if the lateral force is high enough, the vehicle may begin to tip to one side and roll over. **Figure 1** illustrates key roll stability concepts and parameters.

An RS system automatically intervenes if a high-rollover risk is detected while driving. If a rollover threat occurs, the system intervenes and tries to minimize the rollover risk by automatically reducing the throttle and, if necessary, applying the engine and foundation brakes—without driver action. Some RS systems may use drive axle, trailer axle, and steer axle-braking control to reduce speed and regain stability. RS systems are typically integrated with antilock braking system (ABS) controllers, but some may be integrated with an electronically controlled braking system (ECBS).



Vehicle Rollover Due To Excessive Lateral Force (Flat)
(For illustration only – not to scale.)
CG
F_{lat}
R
G
h_{CG}
F_{fr}
F_{fr}
F_{fr}
F_{fr}

Figure 1: Key Roll Stability Parameters

B. Yaw Stability Event

A yaw stability event can occur because vehicles have a tendency to spin about their vertical axis when there is a loss of traction. This loss of traction can be caused by weather conditions or excessive speed in a curve. This can cause loss of directional control, resulting in the tractor and trailer moving along separate paths. When drivers attempt an extreme maneuver or misjudge a curve's severity, they may experience unfamiliar vehicle handling characteristics as the vehicle nears the limits of road traction. If the friction at the tires is not sufficient to oppose lateral forces, one or more of the tires can slide and the result is a loss of vehicle control. This loss of control usually results in either the front of the vehicle "plowing out" or the rear of the vehicle "spinning out."

Figure 2 illustrates the understeer situation in which the vehicle enters a turn, the front of the vehicle slides or plows out, and the vehicle proceeds to the edge of the curve leading to a roadway departure, if uncorrected. **Figure 3** illustrates the oversteer (spinning out) situation in which the tractor-trailer enters a turn, the rear of the vehicle slides or spins out, and the vehicle proceeds toward the edge of the curve. If uncorrected, this situation can lead to a roadway departure, jackknife (for combination vehicles), or other conditions such as tripped rollovers.

C. Electronic Stability Operation

ES systems are active systems that automatically intervene when there is a high risk of roll over or yaw instability. In currently available systems, the ECU constantly compares the vehicle's actual movement to internally stored performance models using the

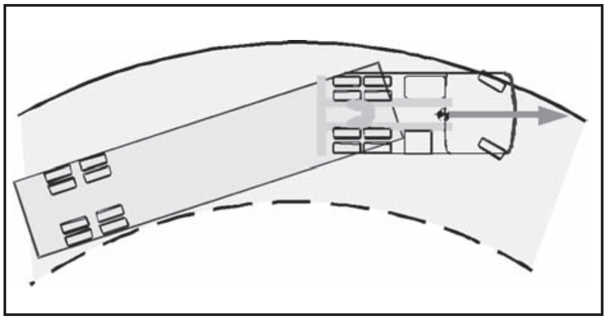


Figure 2: Understeer Condition

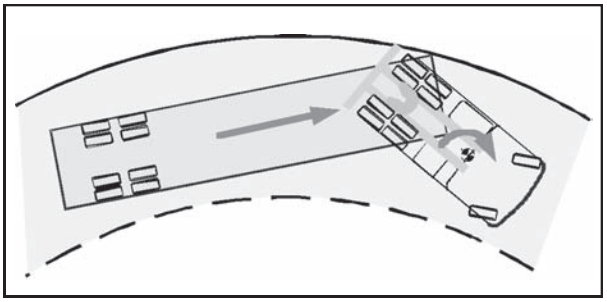


Figure 3: Oversteer Condition

wheel speed sensors, as well as lateral, yaw, and steering angle sensors. If the vehicle shows a tendency to leave an appropriate travel path, or if critical threshold values are approached, the system will intervene to assist the driver.

When a potential rollover risk is detected, the ES reduces throttle and applies the proper brake pressure to slow the vehicle below the rollover risk threshold. When a vehicle slide (oversteer or understeer) is detected, the ES removes the throttle and then selectively applies the appropriate individual brakes as a counterforce to better align the vehicle with an appropriate path of travel. In an oversteer situation, the system applies the "outside" front brake; while in an understeer condition, the inside rear brake is applied. Trailer brakes may also be applied during these situations. These systems may be integrated with ECBS, ABS or conventional pneumatically controlled brakes.

OVERVIEW OF UNDERLYING TECHNOLOGY

The underlying technology of stability systems can be broken down into three primary areas: sensing, programming, and output.

Sensing technology detects potential stability risk quickly and completely. it employs the following sensors:

- **Lateral force or acceleration—**This sensor measures the vehicle's lateral acceleration. The sensor may reside within an ECU or may be mounted separately.
- **Yaw rate sensor—**This sensor measures the vehicle's yaw rate. This sensor may reside within an ECU or may be mounted separately.
- **Steering angle sensor—**This sensor measures the vehicle's steering angle. It typically resides within the steering column.
- **Brake demand sensors—** These sensors measure the driver's brake application. The stability system can use this information to supplement the driver's application throughout the stability maneuver.
- **Load sensor**—This sensor provides the stability system with data about the load on an axle or set of axles. The stability system uses this information to adjust braking pressure or adjust the estimated vehicle weight. The sensor typically attaches to the suspension.

Programming technology is used to determine when the system should intervene based on the central processing unit and parameter programming.

- **Central processing unit** —The stability system onboard computer gathers sensor data and calculates key parameters used to determine the vehicle's likelihood of rollover, oversteer or understeer situation.
- **Parameter programming**—The static and dynamic properties of the vehicle can be loaded into the CPU to help fine tune system intervention in response to the vehicle's likelihood of rollover, oversteer or understeer.

Output technology is used during an intervention to reduce a vehicle's stability risk. It directly affects:

- **Engine throttle control**—Systems may dethrottle the engine if a rollover, oversteer, or understeer risk is detected.
- **Foundation brakes**—Systems may activate the foundation brakes if a rollover, oversteer, or understeer risk is detected.
- **Transmission retardation**—Systems may use the vehicle transmission to apply vehicle control actions.
- **Engine brake**—Systems may activate the engine brake (if the vehicle is so equipped) if a rollover, oversteer, or understeer risk is detected.
- **Visual status indication**—The stability system should provide a visual indication of system status. System status includes operational/non-operational and system fault conditions similar to the manner in which ABS indicators are currently used.
- **Vehicle network**—Stability systems may use the in-vehicle data network (SAE J1708 or J1939) for data communication to data recording or diagnostic devices.

OPERATIONAL ISSUES

A. System Specifications

1. Fleet Management

- All settings for the specific make and model of

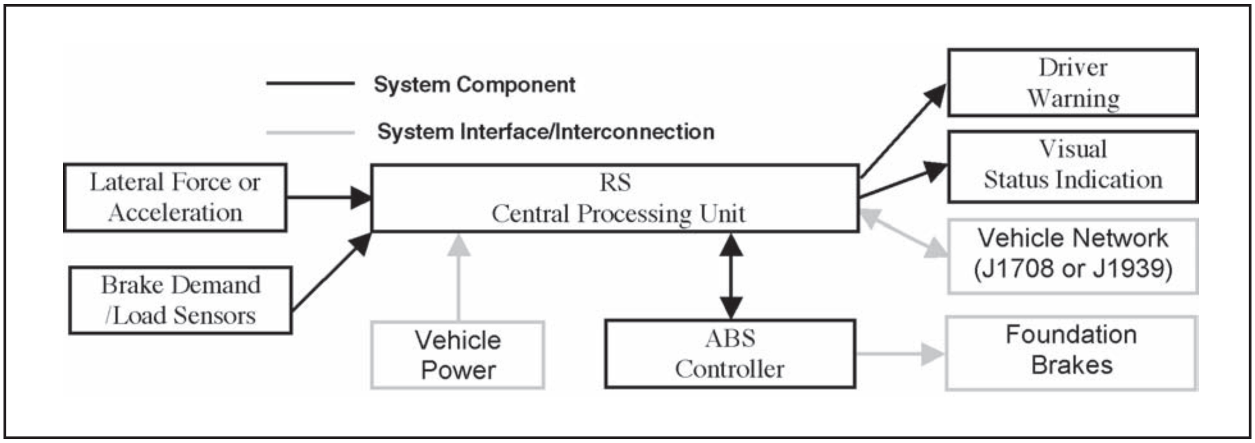

System Component
System Interface/Interconnection
Lateral Force or Acceleration
Brake Demand /Load Sensors
RS Central Processing Unit
Vehicle Power
ABS Controller
Driver Warning
Visual Status Indication
Vehicle Network (J1708 or J1939)
Foundation Brakes

Figure 4: RS Major Functional Components

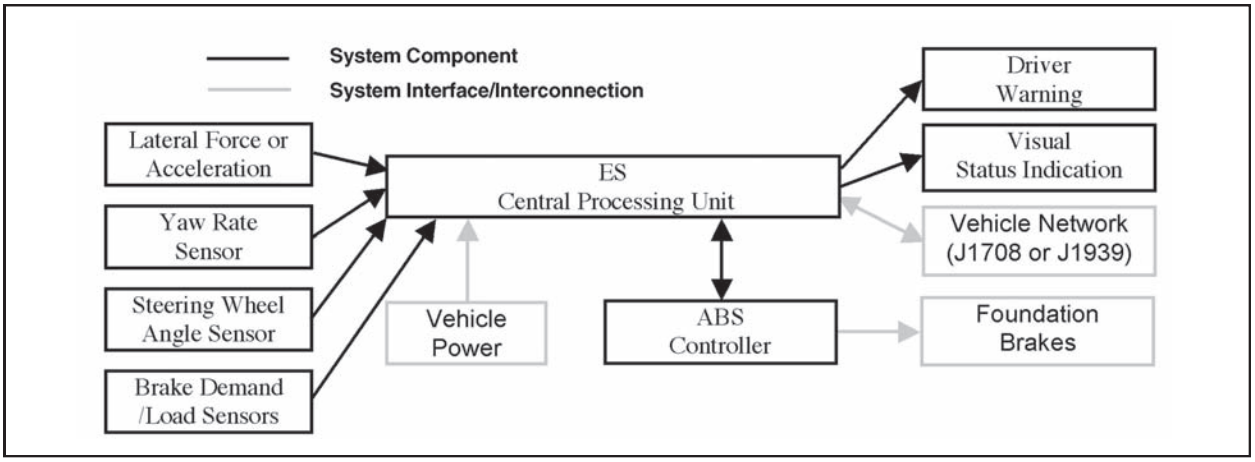

System Component
System Interface/Interconnection
Lateral Force or Acceleration
Yaw Rate Sensor
Steering Wheel Angle Sensor
Brake Demand /Load Sensors
ES Central Processing Unit
Vehicle Power
ABS Controller
Driver Warning
Visual Status Indication
Vehicle Network (J1708 or J1939)
Foundation Brakes

Figure 5: ES Major Functional Components

the vehicle should be supplied to the installers, or the stability system should come preconfigured for use with the vehicle.

- All stability system cables, connectors, and components should be rated for the heavy-duty truck industry as defined by TMC and SAE and be appropriate for their operational environment (i.e., stability system components mounted on the exterior of the vehicle should be rated for exterior duty).
- Major stability system components, excluding cabling or small mounting components, should be marked with manufacturer's identification.
- Stability system in-cab components should remain securely mounted in the event of a crash.
- The embedded software in a stability system may be field upgradeable via the in-vehicle network connection (i.e., J1587 or J1939) or other common data interface (i.e., RS-232 or USB).

2. Driver Adjustments

- No driver adjustments are needed when using vehicle stability systems.
- Drivers should drive in a normal, prudent manner for the driving conditions at hand.

B. Driver Interface Specifications

The following requirements define specific ways in which stability systems interact/interface with the driver, and include indicators, displays, and warning methods. The National Highway Traffic Safety Administration's (NHTSA) Federal Motor Vehicle Safety Standard (FMVSS) 121 should be used as a guideline for stability system indicators.

- Stability systems should indicate to the driver that they are fully operational. This indication may be performed by momentarily lighting a status indicator lamp during power-on system check, similar to the ABS indicator currently used on commercial vehicles and trailers.

- Stability systems should indicate a system failure by continuously lighting a stability system indicator, similar to the ABS system indicator currently used on commercial vehicles and trailers.
- Warnings generated by the stability system should be clearly distinguishable from indicators generated by other vehicle systems.
- Stability systems should not have an on/off or disable switch.
- An indication or label should be supplied with the vehicle notifying the driver that the vehicle has been equipped with a stability system.
- Stability system warnings, alerts, and messages should be readily understood by the driver and not interfere with the driver's primary duty of operating the vehicle.
- Stability system indicators should be clearly legible in direct sunlight and should not distract the driver in darkness.
- Stability systems may provide a visual, audible, or tactile warning to indicate a vehicle rollover is imminent or control actions are taking place.
- Stability systems may audibly indicate a system failure or component malfunction.
- Stability systems may allow the audible warning volume to be adjusted, but not turned down completely.
- Stability systems may provide diagnostic messages such as "Sensor Failure" on an alphanumeric display to alert the driver of specific problems or concerns. See TMC RP 1211, *Electronic Dash Display Guidelines*.

C. Interface with Other Vehicle Systems

Stability systems are designed to interface with other vehicle systems such as engines and transmissions. The vehicle manufacturer and stability system supplier must verify proper operation through vehicle integration testing.

Stability systems typically do not interface with systems such as adaptive cruise control (ACC) and collision warning systems (CWS). The vehicle manufacturer and stability system supplier must verify proper operation through vehicle integration testing. Fleets adding systems such as ACC and/or CWS to vehicles containing Stability Systems should contact the vehicle manufacturer for guidance.

INSTALLATION ISSUES

Stability systems are primarily added to a fleet through new vehicle purchases. Because the systems use additional sensors and components, retrofit installations are difficult. Stability system installation and validation also requires approval of the vehicle manufacturer and the stability system supplier. Depending on the manufacturer's design, some systems may be able suitable for vehicle retrofit through ECU, wiring harness and pneumatic (i.e., valves, lines, fittings, etc.) modifications.

MAINTENANCE ISSUES

The following maintenance and support features/specifications will help ensure proper stability system maintenance and operation.

- Stability systems should require no more maintenance than currently available ABS requires.
- Stability systems should automatically maintain calibration.
- Manufacturers should provide a procedure — via an installation/maintenance manual and PC-based diagnostic software—for recalibrating the stability system after component replacement.
- Manufacturers should provide users with a manual and/or training material for stability system maintenance and operation.
- The user's manual should describe the minimum vehicle speed at which the stability system operates. It should also describe how the system functions and list the various rollover, oversteer, and understeer risk situations that the system can and cannot help mitigate.

Stability systems are validated for a vehicle's original configuration. Contact the vehicle manufacturer or stability system supplier for guidance if the vehicle's chassis components are altered (i.e., a wheel base extension or reduction, tag axle addition or removal, a major body change such as conversion of a tractor into a truck, or an axle, suspension, or steering system component modification). Optionally, the stability system may provide blink codes, observed at the system status indicator, that indicate various stability system faults. Video, audio, or computer-based training material may also be provided for fleet management and/or drivers as an option.

SYSTEM EVALUATION RESULTS

A. USDOT Field Operational Tests

In recent years, the U.S. Department of Transportation sponsored an intelligent vehicle initiative and entered into agreements with four partnerships to conduct field operational tests (FOTs) of advanced intelligent vehicle safety systems. As a part of these activities, both RSA and RS systems were evaluated. The independent evaluation concluded that stability systems operating "under conditions similar to those observed in the FOT are expected to prevent 20 percent of rollover crashes caused by excessive speed[2]".

It is important to note that the RS system evaluated in the testing did not have the capability to apply the foundation brakes (stability interventions were limited to throttle reduction and retarder application). The ability of a RS system to apply the foundation brakes is a key element in its potential performance. It is very likely that current RS systems (with foundation brake application capability) would have an increased level of performance when compared to the system used in the FOT.

B. European Testing

Stability systems for commercial vehicles first appeared in Europe as part of ECBS and have been in production for several years. Manufacturers for the North American market have leveraged European experience in developing these systems.

QUESTIONS FLEETS SHOULD ASK SUPPLIERS

- What type of driving scenarios is the stability system designed to address?
- What types of sensors are used by the stability system and what are their purposes?
- Where are sensors located?
- How does the system get configured for a particular vehicle?
- What type of data does the system store or broadcast? Is the system approved for use with the types of vehicles used by the fleet?
- Will the system work with all relevant components that fleet specifies?
- What financial benefit does the stability system provide?
- What processes/tooling/software required to recalibrate the system when needed?
- What situations require system recalibration?
- What type of equipment is required to troubleshoot the system?

[2] *Reference: "Evaluation of the Freightliner Intelligent Vehicle Initiative Field Operating Test – Final Report – by Battelle, 505 King Avenue, Columbus, OH for the U.S. Department of Transportation, Washington, D.C., 20590. Contract NO. DTFH 61-96-C-00077. Work Order 7718. September 2003."*