TECH/OPS

SEVCON

Gen4 Application Note

Title	Vehicle Setup and Configuration
Filename	App Note - Vehicle Setup and Configuration.docx
Date of Creation	27/11/2008 09:34:00 by Paul Shipley
Last Updated	18/06/2013 09:17:00 by Paul Shipley

Contents

Sevcon Ltd
TVTE
Gateshead
Tyne & Wear
England NE11 0QA

Tel +44 (0)191 4979000
Fax +44 (0)191 4824223
www.sevcon.com

1. Introduction

1.1. Overview

Basic installations follow broadly the same pattern:

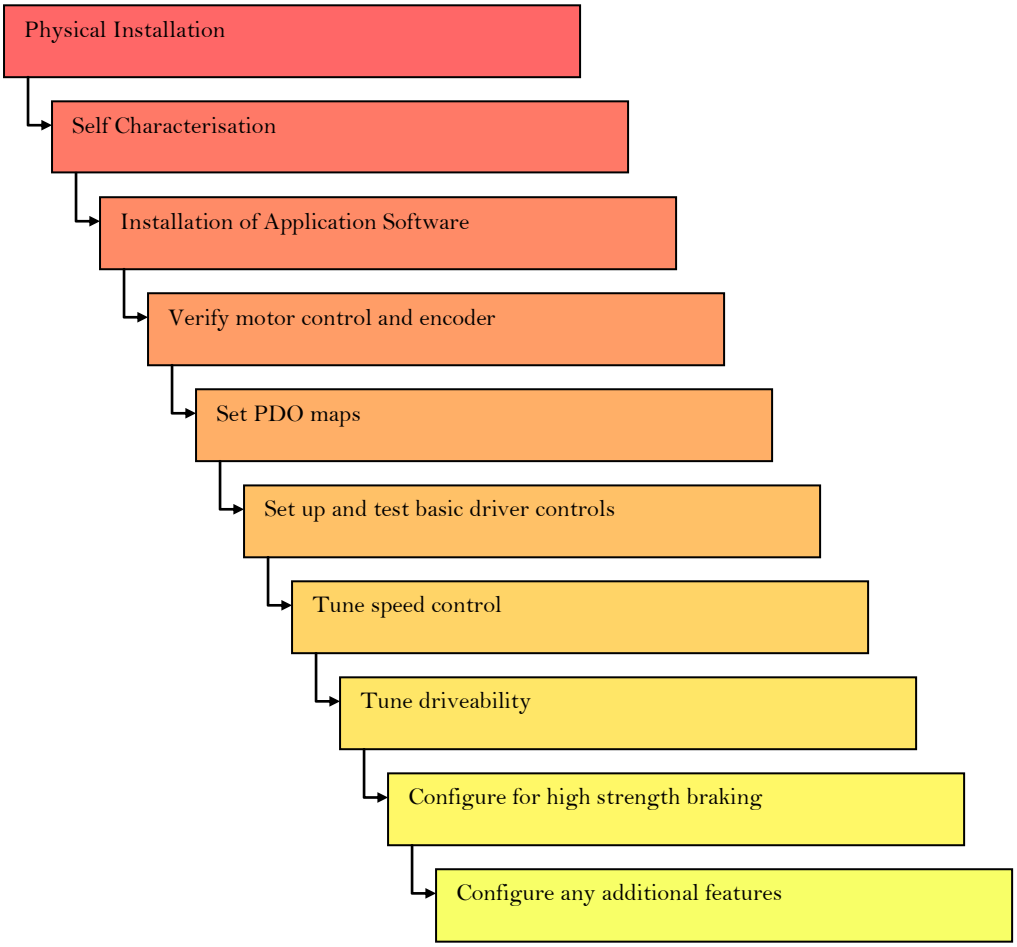

Figure 1 - How to set up a controller in 10 easy steps

In most cases it is difficult to progress to the next stage unless the all previous stages have been completed. For instance, any attempt to tune driveability will be made considerably more difficult, if not impossible, if the speed control is not working.

This application note will provide an outline as to what is involved in each step, what the objectives are, and attempts to highlight any potential pitfalls that may arise.

1.2. Important Safety Information

For those that don't read manuals – at least read this section!

Electric vehicles contain dangerous voltages and currents, and can cause severe damage to property and individuals if not handled correctly. The following words, taken from user manuals, are appropriate:

```
Electric vehicles can be dangerous. All testing, fault-finding and adjustment should
       be carried out by competent personnel. The drive wheels should be off the floor
       and free to rotate during the following procedures. The vehicle manufacturer's
       manual should be consulted before any operation is attempted.

The battery must be disconnected before replacing the controller or one of its fans.
       After the battery has been disconnected wait 30 seconds for the internal
       capacitors to discharge before handling the controller.

Never connect the controller to a battery with vent caps removed as an arc may occur
       due to the controller's internal capacitance when it is first connected.

As blow-out magnets are fitted to contactors (except 24V) ensure that no magnetic
       particles can accumulate in the contact gaps and cause malfunction. Ensure that
       contactors are wired with the correct polarity to their power terminals as
       indicated by the + sign on the top molding.

Do not attempt to open the controller as there are no serviceable components. Opening
       the controller will invalidate the warranty.

Use cables of the appropriate rating and fuse them according to the applicable
       national vehicle and electrical codes.

Electric vehicles are subject to national and international standards of construction
       and operation which must be observed.
```

When setting up a new installation, the vehicle should be immobilised until the basic driver controls have been configured and the driver is able to stop the vehicle safely. Before this point, it is possible for full torque to be output from the motor, regardless of the driver demands.

Until then, the vehicle should be immobilised by either disconnecting the line contactor coil, or raising the drive wheels off the ground. Raised vehicles should be secured with blocks or axle stands. Do not rely on a jack or crane.

Electrical wiring standards must be observed.

1.3. Differences between Gen4 and espAC

The setup process for espAC and Gen4 controllers is mostly identical. The control methods employed by the two controllers are the same, and apply equally to either variant. This document can be used to help set up either of these types of controller.

Differences do occur between the overall featuresets. For instance, Gen4 does not offer as much I/O as espAC, but is able to handle encoders with both 5V and 10V pullups. Nevertheless, the core motor sub-systems both strive for the same ideals with very similar methods of configuration.

Below is a list of major differences. Any other more subtle details pertaining to particular points will be highlighted at the relevant stages in this document:

- Gen4 stores motor electrical characteristics at object 0x4641, whilst espAC uses 0x6410. The objects were moved at the start of Gen4 development as the CANopen standards no longer permitted manufacturer specific objects in the motor profile area (objects 0x6000 to 0x67FF). However, the data stored at this location is used for the same function.
- Similarly, closed loop speed control gains have moved from object 0x60F9 to 0x4651.
- The espAC motor control profile does not support torque control with speed limit. Traction applications requiring torque control should use the Gen4 controller. Basic torque mode without speed limit is available for test purposes.

2. Installation Procedure

2.1. Physical Installation

In this step we install the controller and power it up by the key switch only.

2.1.1. Initial Wiring

Most installations begin with fitting a controller to a vehicle that already has a motor and battery in place, and most of the signal wiring. Wiring for a basic single traction controller can be found in the user manual.

Practically all of the wiring can be installed at the same time as the controller. However, it is advised at this point not to make the final connection to the line contactor coil. Leave one of the spade connectors off. This will mean when the vehicle is initially powered up there will be no chance of unexpected vehicle movement or shoot through.

When the unit is being wired up, it is a good idea to check there are no internal MOSFET short circuits. This can be done when the controller is powered up through the key switch input, but the motor has not been connected yet. To check, simply measure the voltage between B- and each of the motor terminals. It should be around 8-10V. If it is close to 0V or close to battery voltage, then there is a MOSFET short circuit and the unit should not be used.

2.1.2. Special Considerations

Issues regarding the physical and mechanical aspects of the installation are in general beyond the scope of this document, but there are a few factors that should be taken into account:

- Encoder wiring is critical to the operation of the controller. In past applications we have found that several performance issues were caused by poor encoder wiring. Encoder wires should be shielded against noise, be as short as possible, and be routed away from the motor cables.
- Power terminal fixings should be tightened to the correct torque. Over tightening of bolts increases stress on the power board, and in extreme cases can remove the power terminal from the board altogether. The specified torque is currently 11Nm, but this is under review.
- The terminal marked FS is a dummy terminal and does not have any internal connection. If a fuse is mounted to the espAC, the power cable should be connected to the FS terminal so current runs through the fuse and in through the B+ terminal. A common error is to connect the power cable directly to the B+ terminal, therefore bypassing the fuse.
- Fuses are designed to prevent electrical fires starting on the vehicle. All electrical wiring should be protected by appropriate size fuses. Key switch and signal wires should be protected by a 10A fuse.
- CANbus wiring should have correct termination at each end, and stubs should be as short as possible. See the CAN application note for more information.

2.2. Self Characterisation

Self Characterisation software should be installed and executed as per the application note ‘Using SCwiz.doc’.

The characterisation process results in the production of a DVT script. The script will load characterisation values into the objects relevant to the control scheme in use. Note that whilst the self characterisation software does not change any values stored in the controller, the configuration script should be stored and will be used to write the values to the controller's NVM in a later step.

2.3. Installation of Application Software and Settings

2.3.1. Flash Programming

Application software should be installed on the vehicle once Self Characterisation is complete and results obtained. DriveWizard should be able to flash program the application software.

Before continuing, you should ensure that you are familiar with the following:

- Basic CANopen communication principles, particularly the ability to communicate with the controller using the SDO protocol. This will aid debugging and fault finding significantly.

- If programming an espAC, you should be aware of the difference between the 16F and 32F builds of software, and make sure you are programming the correct type. See the application node "256K Host Processor.doc" for further information.

Additionally, you should also be familiar with the basics of CANopen, particularly concepts such as the Object Dictionary and DCF files. Further information can be found in the CAN and CANopen application note.

If this is the first time application software is being programmed into the unit, then special consideration must be paid to the contents of the EEPROM. It is likely that the application software will not work if invalid data is not installed in the EEPROM. If this is the case and you have a new unit, you must 'format' the EEPROM by loading in a blank image using the bootloader. The hardware version and serial number must also be set. See the bootloader application note for further information.

Normally the formatting process and assignment of hardware version and serial numbers is done by the test rigs in production, and should not be a problem. However, if you are working on a virginal unit that has not gone through the full production programming and testing process, you should take this into account.

If the controller is continually crashing[1] then this could be a symptom of invalid data being present in the EEPROM, possible problems during flash programming, or incompatible host and DSP software being programmed. For instructions on how to resolve the situation, refer to the bootloader application note.

You can verify the application code has been programmed by reading out the software version. The software version is available from the Object Dictionary at location 0x100A, 0. The software version should appear to be that of the application software you have just programmed. Note for espAC, the full software version for both the host and DSP builds should be received i.e. "Hxxxx.xxxx Dxxxx.xxxx" or "Txxxx.xxxx Dxxxx.xxxx".

2.3.2. Configuration

Once the application software is installed, you can begin configuration. Controllers that have been flash programmed but do not have a valid configuration will usually power up with a number of faults being set, most likely parameter range faults and sequence faults. The LED will be displaying a flash fault code, much like those on the PowerpaK and MillipaK controllers. This is normal at this stage.

The recommended steps for the initial configuration are to load in the DCF for a similar application, which should set up the entire controller, and then enter the settings obtained from Self Characterisation. The result is you have a controller configured to be the same as the one you took the original DCF from, but has a different motor setup configuration.

It is possible that even after configuration there are a number of other faults set. These should be cleared as much as possible. It is beyond the scope of this document to list corrective action for every single fault, but many of the fault messages should be familiar to those who have used PowerpaK or MillipaK products in the past. It is possible to retrieve a lift of faults that are active on the controller via DriveWizard or the DVT. See the application node 'Fault Diagnosis' for more information.

At the end of this stage the controller should close a line contactor when keyed on. However, since we have not confirmed the vehicle operates correctly yet, the drive wheels should remain raised off the ground.

2.4. Verify Motor Control and Encoder Functionality

At this point we should verify the basic motor control functions and encoder wiring are functional. It is possible using the DVT to control the torque demand being fed to the motor control directly, overriding demands being fed in from the Traction Application.

The techniques shown here are useful to show the basic motor control is working correctly. Without this, it is very difficult to set up speed control loops and driveability profiles.

Before starting, it helps to be familiar with CAN and CANopen concepts, and also VPDOs. Knowledge of how to use the DVT is essential. See the relevant application notes for further information.

There are also certain safety implications that must be considered. Remember the motor is being controlled by the PC, therefore driver controls will not work. **Vehicle drive wheels must be raised off the ground, and secured using axle stands or blocks.** Do not rely on lifting equipment alone to secure the vehicle.

[1] Crashing espAC controllers can be identified by their LED activity. If the LED output is one long flash followed by a short flash, the software has crashed beyond recovery. It will remain in this state until hardware reset (key off then back on). Unfortunately, due to an error on the TI DSP silicon, a similar function to identify crashing Gen4 controllers does not work. However, these can still be identified by repeated transmission of CANopen bootup messages.

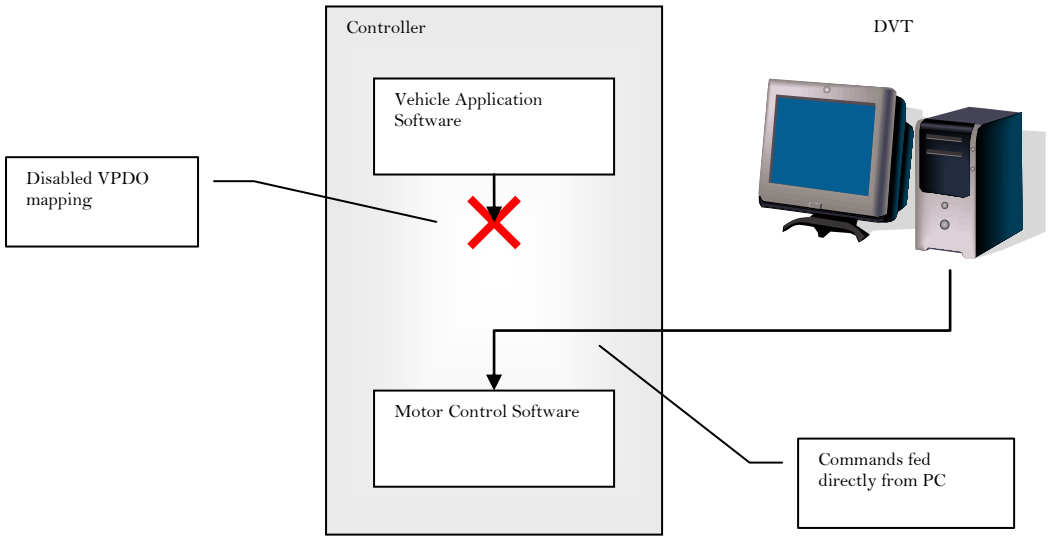

Figure 2 - Motor control commands from the vehicle application software can be disabled, and the motor controlled directly from the DVT

2.4.1. Disabling the Vehicle Application Software

To start, the controller must be in the pre-operational mode. Pre-operational puts the controller into a configuration state and allows us to change the PDO maps in CANopen.

To do this from the DVT you must first log in with the lg command, and go to pre-operational with the fpo command. Note: for simplicity, all the DVT commands in this section refer to node ID 1. If the node ID is different, the appropriate number should be changed.

```
login 1
fpo 1 PRE
```

The line contactor should drop out.

The next step is to remove any PDO mappings that might be feeding control commands to the motor control software. In a normal single motor traction system, there will usually be a single VPDO map from the Traction Application software. It can be removed with the following command:

```
Sdo_ro 1 0x6026
fpo
```

On more complex systems, there may be other maps. The exact intricacies of how these PDO maps work is covered in more detail in the "PDO fundamentals.doc" application note. However, the following sequence of DVT commands can be used to clear any existing PDO maps:

```
source pdo_config.tcl
clear_all_pdos 1
```

Next we must select whether we want to run the motor control in speed control or torque control mode. Vehicle applications generate speed demands. In the initial stages we will need to run the controller in torque mode, to remove any complications caused by the speed control loop.

Speed mode or torque mode is selectable by writing to object 0x6060. Write the value 3 for speed mode and 4 for torque mode. The following command will set up node 1 for torque mode.

```
sdo_wnx 1 0x6060 0 0x04
```

Finally, for Gen4, set the speed control gains to zero to disable any speed limit functionality.

```
sdo_wnx 1 0x4651 1 0x0000
```

```
sdo_wnx 1 0x4651 2 0x0000
sdo_wnx 1 0x4651 3 0x0000
sdo_wnx 1 0x4651 4 0x0000
sdo_wnx 1 0x4651 5 0x0000
sdo wnx 1 0x4651 6 0x0000
sdo_wnx 1 0x4651 7 0x0000
```

To go back to the operational state, enter the following command:

```
fpo 1 OP
```

The line contactor should now close. It is recommended to cycle the key switch at this point. The line contactor should open and then close again. Drive functions will be disabled and the motor will be under the control of the DVT.

If the vehicle has an electrobrake fitted, it may be necessary, depending on the arrangement of the gearbox, to manually release it by powering from an external supply.

2.4.2. Controlling the Motor in Torque Mode

At this point the controller should be configured so that we can send torque demands from the PC for the controller to act upon. This is the most basic configuration we can have. The speed control loops and driveability profiles are not used.

First key on the controller so the line contactor closes. Using a voltmeter, measure the voltage between B- and each motor terminal. There should be approximately 8-10V on each terminal.

Next, enter the following commands[2] in the DVT. These will enable the bridge in the controller and activate the motor control.

```
sdo_wnx 1 0x6040 0 0x0006
sdo_wnx 1 0x6040 0 0x000f
```

You should now see an 8kHz PWM at each motor terminal with approximately 50% duty cycle. If measuring with a voltmeter, you should see approximately half battery voltage between B- and each terminal. It won't be exactly half though, as the controller will be attempting to control magnetising current at this point.

The next step is to set a maximum torque. This is a value from 0-1000, where 1000 represents the maximum output of the motor. This can be set to any value, but setting to 1000 will effectively remove this limit.

```
sdo_wnx 1 0x6072 0 1000 2
```

Note the slightly different format of the sdo_wnx command. As we have specified the value to write in decimal as opposed to hex, we need an additional parameter to specify the length of the value we are writing. In this case 2 bytes. See the DVT application note for more information.

This final command will enter a torque demand. Torque demands are written to 0x6071. Again, the range is from 0-1000, but it is advised to start off with small values initially, and build these up slowly. The following commands request 0.5%, 1% and 1.5% torque output respectively:

```
sdo_wnx 1 0x6071 0 5 2
sdo_wnx 1 0x6071 0 10 2
sdo_wnx 1 0x6071 0 15 2
```

The motor should begin to turn and slowly increase in speed. Because only small torques are being requested, it should be easy to stop the motor using the footbrake. Requesting zero torque should allow the motor to free wheel to a stop. Negative torques should make the motor turn in reverse.

If you cannot get the motor to turn, check the following:

- The controller should be in the operational state, and powered up with the line contactor closed and no faults.
- The statusword of the motor control can be read from 0x6041 and should have the following bits set: 0x0027
- The actual torque output can be read from object 0x6077 and should have the same value at that which is being written to 0x6071. If this is different, then it is being limited by either the power limit map, low battery or over temperature.

[2] For interested parties – the first command in this sequence is used to disable the motor control, and the second to enable it. However, CANopen DSP402 specifies that you must issue a disable command before the first enable command. Enter the first command again to disable the bridge if required in future.

If when requesting positive torque the motor turns slowly forward and is difficult to stop, then you are likely to have encoder problems. See the next section.

If when requesting positive torque the motor turns slowly in reverse and is difficult to stop, then the direction of rotation is wrong. You can correct this by swapping over any two of the motor cables connections.

If when requesting positive torque the motor turns smoothly in reverse and vice versa, then you need to swap both the direction of rotation and the encoder signals. Swap over any two motor cable connections, and also swap encoder signal A and B connections.

2.4.3. Checking the Encoder

The operation of the encoder is fundamental to many areas of the motor control. Without it, motor control, speed control and driveability will not work.

Checking the encoder is simple. With the motor turning, read the speed from object 0x606c. You should receive a speed in RPM – positive values for forward and negative values for reverse[3]. This test can be performed with the motor control disabled and the line contactor open. Speed measurement runs all of the time and can be checked by turning the motor shaft by hand.

If you receive negative values when the motor is turning forwards and vice versa, then the encoder inputs are inverted. Swap encoder inputs A and B over to correct this.

If you receive speeds that are not valid, then the number of pulses per revolution may be incorrect. The encoder setup is at object 0x6090. Setting sub-index 2 of this object to 1 means that sub-index 1 directly represents the number of pulses per revolution for the encoder.

If no encoder signal is being received at all, then the measured speed will remain at zero. If this is the case, the encoder pull up at object 0x4630 may not be configured correctly. If neither the current sink nor voltage source modes work, then the signals at inputs A and B should be monitored using an oscilloscope to check a quadrature signal is being received, and the encoder is powered correctly.

Normally, SCwiz would have reported encoder errors during the open loop AC test. The encoder setup is also included in the setup script generated by SCwiz. If encoder problems persist, it is recommended to go back to SCwiz and retry the open loop AC test to assist with fault finding.

2.4.4. Controlling the Motor in Speed Mode

Once the motor can be controlled reliably in torque mode, we can go to speed mode. This will bring in the speed control loop. To do this, the system needs to be configured for speed control as explained in section 2.4.1 – with object 0x6060, 0 is set to 0x03.

Note for Gen4 – even if you intend the final system to run in torque mode with speed limit, it is still useful to complete this section and start by driving the vehicle in speed control mode as the same gains can be used with speed limit in torque mode.

To start off, the controller should be in operational and the line contactor closed. We should also set some nominally low speed control gains. The gains go into object 0x60F9, with sub indices 1 and 3 representing the proportional gains, and sub indices 2 and 4 representing integral gains. Start with the integral gains set to zero, and proportional gains set to low values such as those suggested below. (Remember espAC uses object 0x60F9 instead of 0x4651).

```
sdo_wnx 1 0x4651 1 0x0020
sdo_wnx 1 0x4651 2 0x0000
sdo_wnx 1 0x4651 3 0x0020
sdo_wnx 1 0x4651 4 0x0000
```

For more information on how to set up speed control, see the application note on tuning the speed control loop. For now though, it is sufficient to keep sub indices 2 and 4 set to zero, and set 1 and 3 to be the same value.

Controlling in speed mode is very similar to controlling in torque mode. The only difference is we write a speed demand instead of a torque demand. First, activate the bridge:

```
sdo_wnx 1 0x6040 0 0x0006
sdo_wnx 1 0x6040 0 0x000f
```

Then set a value for maximum torque (writing 1000 equates to 100% torque available):

[3] Negative values are often expressed internally as large positive values, where the most significant bit specified the sign of the number. The DVT shows raw values in this format. For example, a signed 32 bit value shown as 0xFFFFFFFC actually represents -4.

```
sdo_wnx 1 0x6072 0 1000 2
```

Also set a maximum value for speed limit (value written is in rpm)

```
sdo_wnx 1 0x6080 0 2000 4
```

Then set a speed demand. Speed demands are written to object 0x60FF, with units being in rpm. A 32-bit number should be written. For example, the following command will request 60 rpm:

```
sdo_wnx 1 0x60ff 0 60 4
```

If you are lucky, you will get 60 rpm from the motor – 1 rotation per second. It is more likely however that the motor will be spinning slower than this, in which case you should increase the proportional gain. If the motor is over speeding, or oscillating violently, the proportional gain should be decreased.

The actual speed of the motor as measured by the controller can be read from object 0x606c.

The intention here is to give enough control to perform simple manoeuvres with the driver controls later on. It is not important to get exact speed control at this stage, and the integral gain should be left set to zero. Repeat the above steps over a range of speeds up to 2000rpm.

The motor should come to a complete stop when zero speed is requested.

2.5. Set up PDO mappings

By now we should have our motor controller set up to a point where we can set speed demands and have the motor turn smoothly at approximately the requested speed. In this step we will set up the PDO maps so we can control the vehicle from the driver controls.

As an alternative to this section, please note than Gen4 also offers auto-configuration functionality, which is especially useful for setting up multi-node applications. For more information, please see the "Autoconfig setup" application note.

2.5.1. Setting PDO maps

It the application consists of only a single node, then the vehicle can be set up using VPDOs only. The following DVT commands can be used to configure a standard set of VPDO maps for a standalone traction controller.

```
# Login and go to pre-operational
login 1
fpo 1 PRE

# Set up the local traction motor
sdo wnx 1 0x3000 0 0x01
sdo wnx 1 0x3000 1 0x2020

# Set up line contactor on first analoge output
sdo_wnx 1 0x3200 0 0x01
sdo_wnx 1 0x3200 1 0x2400

# Set up first 4 digital inputs as fwd, rev, FS1 and seat switches
sdo wnx 1 0x3300 0 0x04
sdo_wnx 1 0x3300 1 0x2121
sdo_wnx 1 0x3300 2 0x2122
sdo_wnx 1 0x3300 3 0x2123
sdo wnx 1 0x3300 4 0x2124

# Set up throttle input on first analogue input
sdo_wnx 1 0x3400 0 0x01
sdo_wnx 1 0x3400 1 0x2220
```

For more complicated setups, you should refer to the application note 'PDO fundamentals' for further information.

Alternatively, Drive Wizard can be used to create the required PDO maps. However, at time of writing, only VPDO maps are supported.

2.5.2. Testing the PDO maps

Before proceeding, you should check at this point that the PDO maps are performing as they should. Each input, output and motor mapping should be checked to ensure the map is functional.

PDO maps can be verified by going to operational and reading the values that are written into the application objects in the Object Dictionary between 0x2000 and 0x24FF. For example, if the throttle is connected to analogue input 2 of node 4, you can read it using the following command:

```
sdo_rnx 4 0x6C01 2
```

The value read in this object should be copied into the throttle input voltage object of the master node. If the master node is node 1, then the following command will read it:

```
sdo_rnx 1 0x2220 0
```

Changes to the throttle voltage input should be copied to both Object Dictionary locations specified above.

This should be repeated for each input and output.

2.6. Check Basic Driver Controls

The objective at this stage is to perform some final checks to ensure we can safely drive the vehicle from the driver controls.

Ensure at this stage that the controller is configured to generate speed demands, i.e. object 0x2900,0 is set to 0x01.

2.6.1. Driver Control Configuration

Configuration of driver controls at this point is similar to that of the PowerpaK. Objects in the range 0x2800 – 0x3000 are most relevant, with many of the objects having recognisable titles.

For a standalone traction controller the following objects will be of most interest:

- Set up object 0x2910 to configure the throttle parameters. The start and end voltages should be set to the range of voltages that are written to object 0x2220 by the PDO maps. For example, you may get 0x0120 at 0% accel push and 0x04B0 at 100% accelerator push.
- The start and end values in 0x2910 are similar to the creep speed and max speed personalities of the PowerpaK. However, rather than a percentage, they are scaled from -32768 to 32767, representing -100% to +100%.
- The accelerator profile is set by object 0x2910, 2. The normal presets are available (linear, crawl, etc…). By setting this to custom, you can define your own map in sub indices 11-16.
- The driveability profile is set up in object 0x2920. At this stage, you can enter modest drivability parameters to get the vehicle moving, particularly light braking. These will be refined further in sections 2.8 and 2.9.

2.6.2. If the Vehicle Doesn't Drive…

If the vehicle is not driving, then some point in the control pipeline is not working correctly. The control pipeline is a sequence of functions the driver demands passes through in order to be converted from a series of electrical signals to 3 AC motor currents.

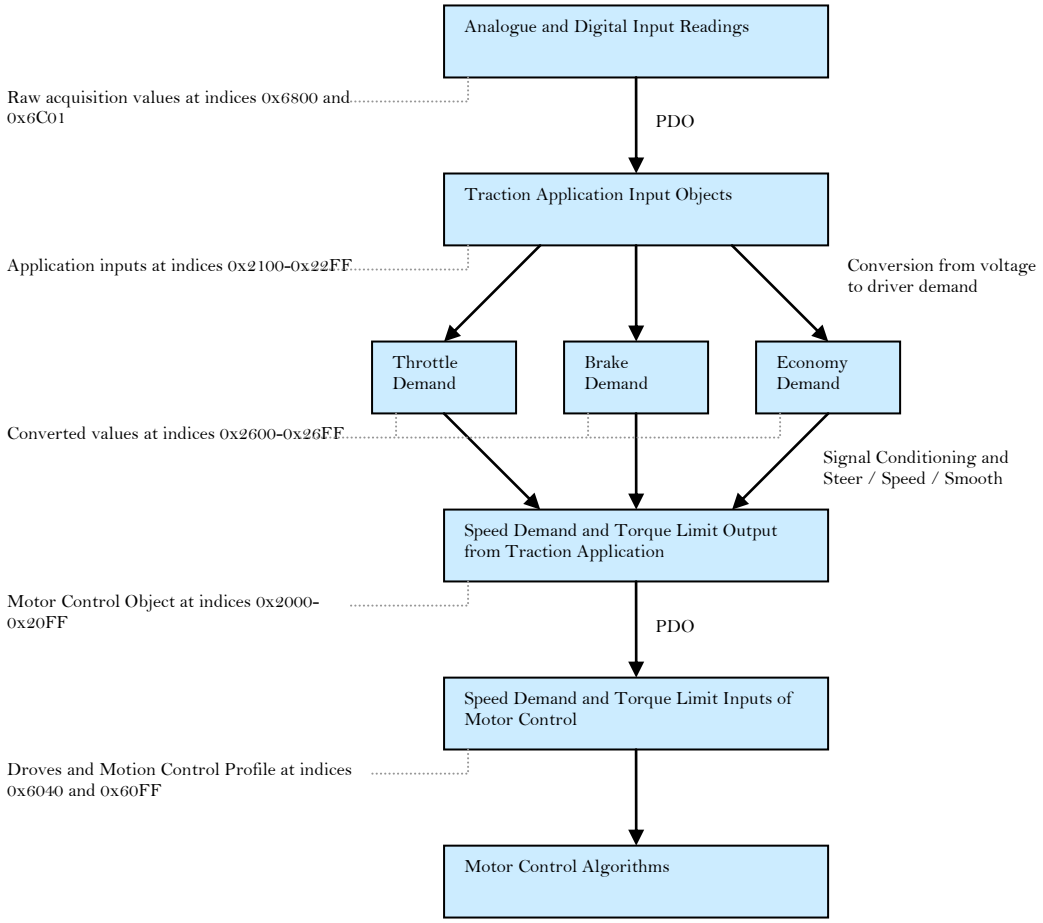

Figure 3 - Typical control pipeline for a standalone traction system

There are a number of points in the object dictionary we can look to see how far through the pipeline the driver demands have got. The diagram above highlights some of the Object Dictionary entries that should be checked when trying to determine why a vehicle does not drive when expected.

- First, check the inputs are being read correctly by the Generic I/O Profile objects. These are all located in the range 0x6800 – 0x6FFF. Typically, you would have a throttle input and a number of digital inputs. The values at objects 0x6800 and 0x6C01 should change according to changes in driver demands. If they do not, this indicated an electrical wiring problem.
- The values in the Generic I/O Profile should be copied into the vehicle application input objects. These are located around 0x2100-0x22FF. If you do not see the changes in driver demands reflected in these objects, then your PDO mappings are not working[4].
- The vehicle applications will convert an input voltage into a driver demand. This is similar to the 'push' test items on PowerpaK and MillipaK controllers. However, instead of being given as a percentage, they are given as a signed 16 bit number, ranging from -32768 to 32767. If you do not get the expected output, check the scaling objects at 0x2910-0x291F.
- In addition, check for any inputs that you are not expecting to be mapped in at this point. For example, you may have a PDO map for footbraking or economy configured but not realise it, and it could be this that is limiting the vehicle output. The demand values for any unused inputs should remain at zero.
- The driver demand objects will go through some signal conditioning and ramp functions. For traction applications, these can be fairly complex and are effectively given their own subsystem called Steer/Speed/Smooth. If you can see driver demands, but fail to see and demand from objects at the motor control objects at 0x2000-0x20FF, then there are problems with the ramping functions. Check

[4] Remember when debugging this, PDO mappings are disabled when the system is not in the operational state.

the maximum speeds and ramp rates. For Steer/Speed/Smooth, you should also check for any driveability profiles that are active (see object 0x2720).

- If the motor control objects are not being copied into the Drives and Motion Control Profile, then there are further PDO mapping problems.

The above steps give a very brief insight into the process of converting driver demands into control commands suitable for any CANopen Drives and Motion controller. It will give an indication as to which part of the process is most likely to be causing problems, but it does not go into such detail as:

- Configuring ramp rates
- Smoothing and conditioning of signals
- Checking sub-system states
- Motor control algorithms

For more information on these subjects, see the Driver Demand Pipeline application note.

2.6.3. If the Vehicle Does Drive…

If you find you are now able to maintain reasonable control over the vehicle powertrain from the driver controls then it is time to put the vehicle on the ground and begin driving.

As a final check, ensure that:

- You have easy access to an emergency stop switch from the normal drive position, and you have tested it to make sure it isolates the controller from the power supply as expected
- Mechanical brakes are in good working order
- The integral gains for the speed control loop are set to zero (sub-indices 2 and 4 in the speed control gain object)
- The vehicle drives correctly in forward and reverse according to the driver controls, and stops when neutral is selected
- You have plenty of space available in front of and behind the vehicle

If you find you were able to drive the wheels when they were raised off the ground, but when the wheels are lowered you find the vehicle struggles to move or doesn't move at all, you can increase the torque demand by increasing the proportional term on the speed loop by a small amount. Actual torque is available at object 0x6077 and ranges from -1000 to +1000. If the actual torque is at maximum value and the vehicle still does not move, the characterisation data should be checked.

2.7. Tuning Speed Control Loops

The speed control loop used in the Drives and Motion Control profile is a standard closed PI loop:

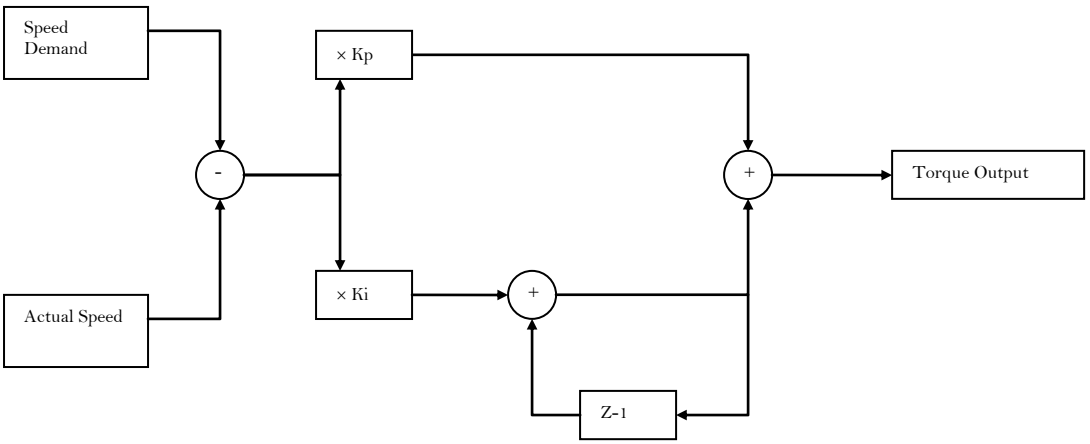

Figure 4 – Overview of closed loop speed control

The speed control loop is considered part of the DSP402 Drives and Motion Control profile, and resides and is executed on the DSP. The same algorithm runs on all controllers; the difference between a traction or a pump motor is not important to the speed control loop. The loop is executed once every 5ms.

For speed loop tuning techniques, refer to the application note 'Tuning the Speed Control Loop'. This will run through the basics of tuning the speed control loop, as well as some advanced techniques for assisting entry to hill hold and overcoming problems associated with slack in gearboxes.

2.8. Tuning Driveability

Vehicle driveability refers to the 'feel' of the vehicle when it is driven. Different customers will have different ideas about how the vehicle should drive, be it with aggressive rates of acceleration and deceleration, or with more gentle rates with little electrical braking assistance. The feel of the vehicle can be set up by adjusting the vehicle driveability parameters.

This section describes setting up driveability in speed control. As mentioned before, Gen4 applications that are intended to run in torque mode with speed limit should still complete this step as the control gains used in speed control will apply to speed limit.

2.8.1. Basic Setup

The output from the traction application is a target vehicle speed. This speed is fed to the target speed Object Dictionary entry of the Drives and Motion Control profile. By changing the rate at which the target speed increases and decreases we can affect how the vehicle feels when driven.

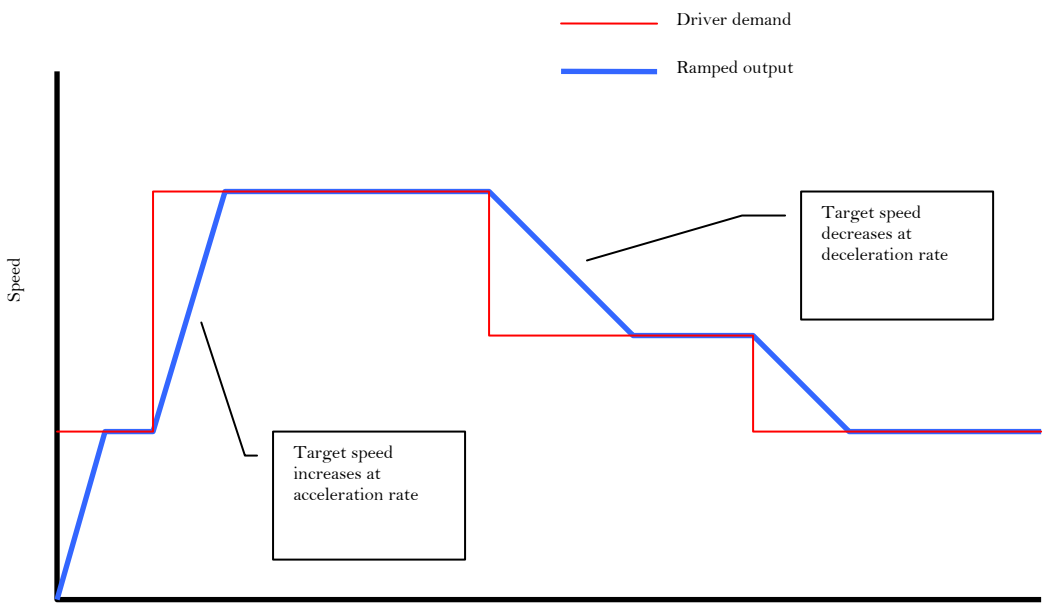

Figure 5 - Classic representation of acceleration and deceleration rates.

There is also the possibility to limit the amount of torque available in any state. For instance, you may wish for full torque to be available in drive, but limit the amount of torque available in braking to give the feel of coasting when in braking.

Driveability is configured in objects 0x2920-0x292F. A number of driveability profiles are available. Normally, the baseline profile at index 0x2920 is used. However, other driveability profiles may come into play when, for instance, the vehicle battery is low, or any of the driveability select switches are closed. In these instances, the lowest value from each driveability object is used.

The format of each of the driveability configuration objects is the same:

Index	Sub Index	Function
0x292X	0	Number of available sub-indices. This is read only and is always set to 14.
	1, 2, 3, 4	These sub-indices set the maximum allowed torque in each of the available states; drive, direction change, neutral braking and footbraking. A value from 0-1000

		should be entered, where 1000 = 100% torque available.
	5, 6	These sub-indices set the maximum speed in forward and reverse. These are the speeds that the target speed will be ramped to when a throttle value of 100% is received from the traction application.
	7, 8, 9, 10	These sub-indices specify the acceleration ramp rates for each available drive state, specified in rpm per second.
	11, 12, 13, 14	As above, but for deceleration rates.

Acceleration and deceleration rates are specified in rpm per second. For example, if the top vehicle speed is set to 5000rpm, and the ramp rate is set to 800 rpm per second, it will take 6.25 seconds for the target speed to ramp up from zero to top speed.

A common question at this point is "why are there acceleration rates for the braking states?". The answer is they were originally used for torque control systems as the rate at which the braking torque would ramp up. They are not particularly relevant to speed control systems, but have been maintained for completeness and may be reinstated if a torque control mode is ever implemented.

At this stage you may find there is not a lot of braking torque available at high speeds. Braking can be left light at the moment, but stringer braking can be configured in section 2.9 below.

2.8.2. Feathering [PS1]

In speed control systems, there will often be a limited amount of overshoot or undershoot when performing a step change to a target value. Normally this is not a problem, and should not be particularly noticeable when driving.

There is however one situation where is can be noticeable – end of braking. In these situations, it is common to experience a level of undershoot which manifests itself as the vehicle driving a small distance in the opposite direction to the last drive direction just as the vehicle comes to a halt.

If we produce a graph of target speed versus actual speed, we would probably see something like the following:

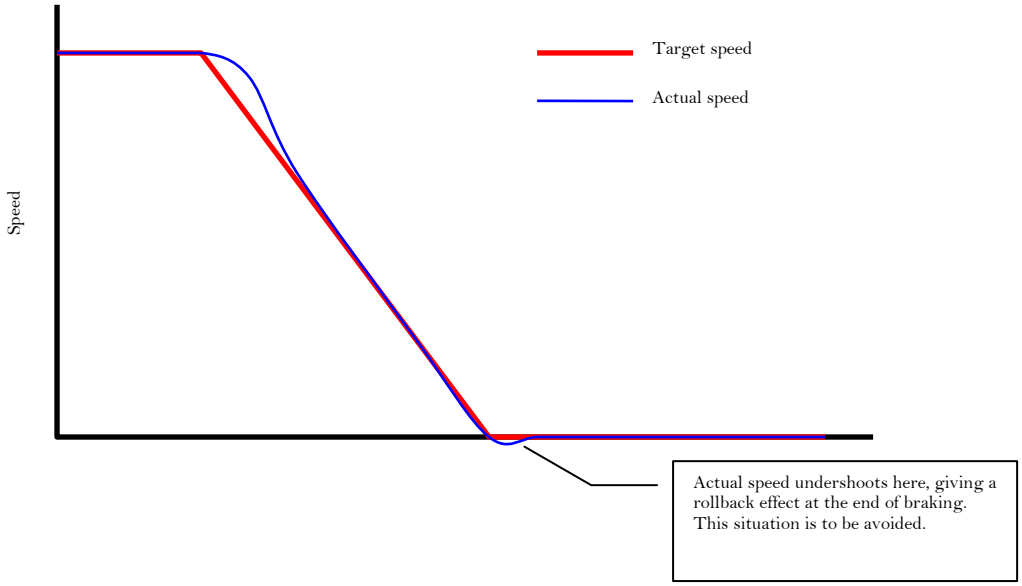

Figure 6 - End of braking without feathering

In order to avoid the above situation, we introduce feathering to try to guide the actual vehicle speed in to zero speed in a more controlled fashion. This is done by reducing the deceleration rate just before the vehicle stops. The above graph now turns into:

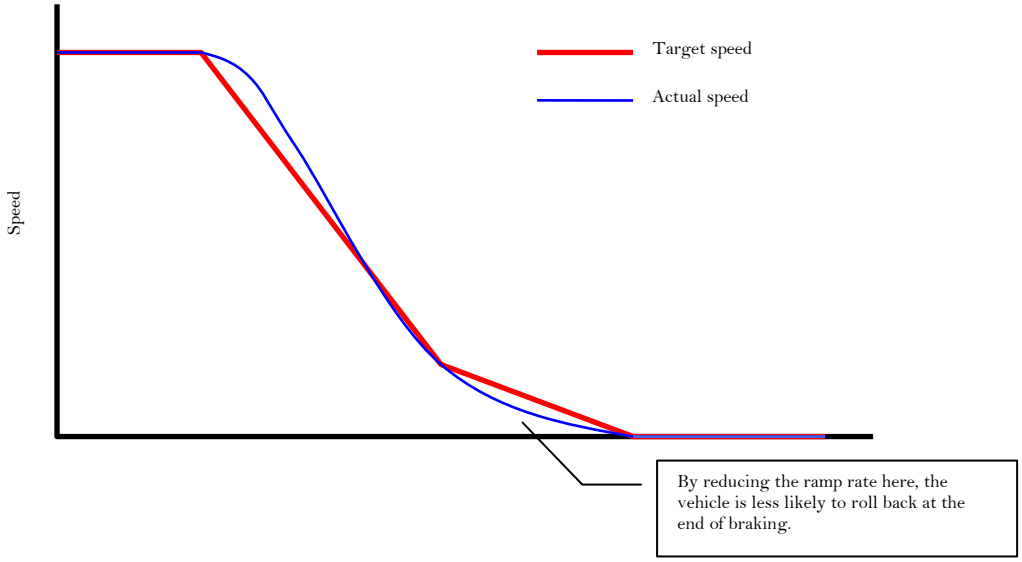

Figure 7 - End of braking with feathering

Feathering is configured in the Object Dictionary at index 0x2905. 2 parameters are required; the speed at below which feathering is active, and the ramp rate to be used during feathering. Typical values would be to have the feathering ramp rate be active when the target speed is below 300rpm, and to reduce the ramp rate to 100 rpm per second.

Feathering will not activate if the motor has to output drive torque to maintain the target speed. This is to avoid the feeling of drive torque being produced during braking when driving uphill. Additionally, feathering will not activate during direction braking.

2.8.3. Special Steer/Speed/Smooth Considerations

There are some instances where the Steer/Speed/Smooth sub-system will override the prescribed ramp rates and insert step changes in demand, or change the ramp rate so the target speed follows the actual speed. However, in most cases these will produce desirable results and enhance the performance of the vehicle.

Two notable situations occur during braking. The first comes into effect if the vehicle goes into braking but the motor cannot produce enough torque to maintain the target speed. In these cases, on entry to braking, the target speed will step to the actual vehicles speed so the braking effect can be felt immediately by the driver i.e. there is no delay while the target speed ramps down to the actual speed.

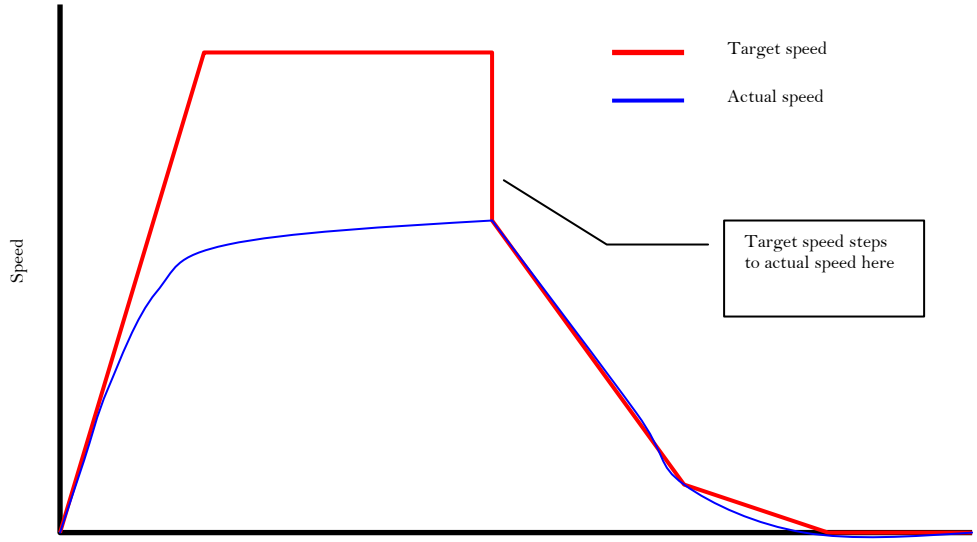

Figure 8 - Target speed versus actual speed on entry to braking where target speed is not being maintained

The second scenario where ramp rates may be changed by steer/speed/smooth is during braking where the actual vehicle speed is decreasing faster than the target speed. This would be due to some external influence, such as the application of the footbrake or because the vehicle is towing a load up a hill. In order to prevent the motor driving against this braking effort, the ramp down rate of the target speed will be increased to it follows the actual speed. In effect, this will produce zero error, and therefore result in an output of zero torque from the speed loop.

2.8.4. Disabling Drive Torque

The traction control software offers the ability to disable drive torque in certain situations where drive torque is undesirable. Usually this means no drive torque will be produced when in any of the braking states.

During footbraking, all drive torque will be disabled – torque may only be applied in the opposite direction to which the motor is turning. Just before the vehicle comes to a stop (actual vehicle speed is below 200rpm), all torque will be disabled. This will clear out any integral gain that may have built up during footbraking, and prevents the vehicle from moving when the footbrake is released.

During neutral and direction braking, drive torque is disabled until the vehicle is below the speed specified in Object Dictionary entry 0x2908. The reason for this is it is important drive torque is disabled during the majority of braking, especially at high speed, but it is useful to allow drive torque to come in at a safe low speed, typically around 400rpm, to assist in entry to hill hold, or to provide drive in the opposite direction.

2.8.5. Additional Options

Driveability profiles allow the vehicle to be configured in a variety of different ways. Some examples would be:

- Use the low speed driveability profile to provide additional low speed manoeuvring capabilities. The low speed profile is active whenever the vehicle speed is below that which is specified in object 0x2906.
- Mount the driveability select switches on the dashboard, configure them as “tortoise and hare” switches, to allow the driver to have the vehicle change between different modes of behaviour.
- A direction change delay can be set at index 0x2909. This can be used to inhibit direction changes for a short while producing behaviour similar to DC vehicles.

2.9. Configuring High Strength Braking

Some applications require more aggressive braking strength than others. Unfortunately, the power limit map associated with closed loop current vector control does not always allow the necessary torque to be produced at high speeds.

The open loop braking technique can be used to increase the amount of braking torque that can be produced. For more information on this, refer to the application note “Configuring Vehicles for String Braking”.

For some applications, you may also wish to reduce the amount of torque available for open loop braking. Sometimes the braking effort is so strong it can cause the vehicle wheels to lock. By limiting the available torque, you will increase the stopping distance for some applications, but without running the risk of locking the wheels.

2.10. Enabling Torque Control with Speed Limit

Gen4 controllers offer an additional control mode where the throttle position sets the motor torque demand rather than the speed demand. So far, this application note has dealt with the vehicle running in speed control mode. The following steps will convert it to run in torque control mode, with the gains that were used in closed loop speed control mode now controlling the speed limit.

First, log in and request pre-operational:

```
login 1
fpo 1 PRE
```

Next, convert tracapp to produce torque demands as opposed to speed demands.

```
sdo_wnx 1 0x2900 0 0x00
```

Each of the slave motor controllers should be reconfigured to run in torque mode.

```
sdo_wnx 1 0x6060 0 0x04
```

The driveability profiles should be amended to produce torque ramps. Objects at 0x2920 should be amended as follows:

Index	Sub Index	Function
0x292X	0	Number of available sub-indices. This is read only and is always set to 14.
	1	Sub-index 1 sets the maximum amount of torque available in drive. For the baseline profile, this should be set to 1000 (equal to 100%). Lower values may be useful in other driveability profiles.
	2, 3, 4	These sub-indices specify the amount of braking torque to apply in direction change, neutral braking and foot braking modes. The amount of braking torque requested for each mode will remain constant. Values of 500 (50%) for direction and foot braking, and a value of 200 (20%) for neutral braking are typical.
	5, 6	These sub-indices specify the speed limits for forward and reverse drive. Values of 4000rpm for forward and 1000rpm for reverse drive are typical.
	7, 8, 9, 10	These sub-indices specify the ramp rates to use for increasing torque. The values are specified in thousandths of peak torque per second. A typical value of 5000 for each of these sub-indices means full torque will be achieved in one fifth of a second. This value can be adjusted to prevent sudden changes in torque demand. Different values can be specified for drive and each braking mode as required.
	11, 12, 13, 14	As above, but for deceleration rates. A typical value would be around 10000, meaning torque will decrease from full to zero in one tenth of a second. Again, different values for different modes can be specified if required, but a high number such as 10000 is recommended here to ensure delays on entry to braking are kept to a minimum.

The speed limit ramp rates at object 0x291B should also be set.

Index	Sub Index	Function
0x291B	0	Number of available sub-indices. This is read only and is always set to 2.
	1	Specifies the rate at which the speed limit value will increase. Specified in rpm per second. Should be set to a high value such as 2000rpm/s to prevent it interfering with normal acceleration performance.
	2	Specified the rate at which the speed limit value will decrease. Again, specified in rpm per second. A typical value of 300rpm/s allows for slow controlled deceleration. The effect of this parameter is most noticeable when performing reduction braking, or when the braking torque levels in the main driveability profile are not strong enough to stop the vehicle.

Finally, return to operational and drive the vehicle. Speed limit should work as before. Graphing vehicle performance with the DVT will show how well the speed limit is being maintained, but some further speed limit tuning may be necessary.

2.11. Additional Features

With the vehicle running smoothly, it is now possible to add some of the finishing touches. Some of the more academic features, such as alarm buzzers, electrobrakes, warning indicators and displays, can be added at this stage.

Anything added at this stage should not affect the performance of the vehicle.

3. Vehicle Commissioning