Example - Profile Velocity Mode

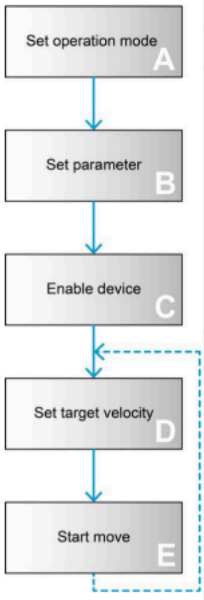
> **Warning**
>
> Running this example turns the motor shaft. Before the first run: secure the motor mechanically, remove or secure anything mounted on the shaft, and keep the power supply within reach. The target velocity is 0 at start; press `+` once (100 rpm) and watch the shaft before going further. `q` switches the power stage off - the shaft then coasts to a stop, it is not braked. If the program is interrupted in another way (closing the window, `Ctrl+C`), the drive keeps the last state it was commanded; switch the supply off if in doubt.

Prerequisites

- An ESCON2 connected over USB, with logic and motor supply switched on.
- Motor and sensor configured and the controller tuned with **Motion Studio**, and Motion Studio **closed** afterwards - it holds the USB port.
- The Motion Library installed and the example built, see Getting Started and Examples.

What the example does

`main.cpp` follows the *Profile Velocity Mode* sequence of the *ESCON2 Application Notes* (table *Device programming - Profile Velocity Mode*, downloads on the product page):



#	Object name	Object	Description [unit]
A	Modes of operation	0x6060-00	0x03 = (Profile Velocity Mode)
B	Max profile velocity	0x607F-00	Motor- and application-specific [rpm]; depending on motor, mechanics, application
	Profile acceleration	0x6083-00	Application-specific [rpm/s]; default: 10'000 depending on electrical and mechanical limits
	Profile deceleration	0x6084-00	Application-specific [rpm/s]; default: 10'000 depending on electrical and mechanical limits
	Quick stop deceleration	0x6085-00	Application-specific [rpm/s]; default: 10'000
	Motion profile type	0x6086-00	Application-specific [0]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Target velocity	0x60FF-00	Desired velocity [rpm]
E	Controlword	0x6040-00	0x000F

Table 2-6 Device programming | Profile Velocity Mode (Start)

Step	Call in `main.cpp`	What happens on the device
Connect	`escon2.Connect("ESCON2", "MAXON SERIAL V2", "USB", "USB0")`	Opens the USB connection
Recover	`TryRecoverFromFault()`	If the device is in 'Fault', requests a fault reset
A	`SetModeOfOperation(OperationMode::PROFILE_VELOCITY_MODE)`	Selects PVM and verifies it
B	`PVM_SetParameters(4000, 5000, 5000, 10000)`	Velocity limit and the three ramps
C	`EnablePower()`	Walks the state machine to 'Operation Enabled'; the motor is now energized
D, E	`PVM_ApplyVelocity(velocity)`	Writes the target velocity and starts the move
Loop	keyboard	`+` / `-` change the velocity, `s` toggles a quick stop, `q` quits
Exit	`DisablePower()`, `Disconnect()`	Power stage off, connection closed

The console shows the averaged actual velocity, the target velocity and whether a quick stop is active:

```
[LOG] ESCON2 connected!
[LOG] Device state: Operation Enabled
[LOG] ------------------------ Terminal Interface ------------------------

(+) to increase by 100rpm | (-) to decrease by 100rpm
(q) to quit               | (s) toggle quick stop

averaged:    500rpm / target:    500rpm
```

Parameters used by the example

Parameter	Object	Value in the example	Unit	Note
Max profile velocity	0x607F	4000 (`MAX_VELOCITY`)	rpm	Must not exceed *Max motor speed* configured in Motion Studio, otherwise `PVM_SetParameters` throws
Profile acceleration	0x6083	5000	rpm/s	Ramp when speeding up
Profile deceleration	0x6084	5000	rpm/s	Ramp when slowing down
Quick stop deceleration	0x6085	10000	rpm/s	Ramp used by the quick stop
Target velocity	0x60FF	0, changed in steps of ±100	rpm	The sign is the direction of rotation

Lower `MAX_VELOCITY` in `main.cpp` for your first runs.

The device state machine in one minute

The ESCON2 follows the CiA 402 device state machine. Only in **Operation Enabled** does it accept a target velocity; everything else in the example exists to get there and back. The states the example passes through, and the controlword commands between them:

```
Switch On Disabled ──«Shutdown»──► Ready To Switch On ──«Enable operation»──► Operation
Enabled
        ▲                                                                          │        ▲
        │                                                              «Quick stop» │        │
«Enable operation»
        │                                                                          ▼        │
        │                                                                      Quick Stop
Active
        │
        ├── «Disable voltage» ── from every state above: power stage off, shaft coasts
        │
        └── «Fault reset» ──────  Fault  (entered from any state when the device detects an
error)
```

Method	Command(s) it sends	Resulting state
`EnablePower()`	«Disable voltage», «Shutdown», «Enable operation»	Operation Enabled (power stage on)
`PVM_ApplyVelocity(v)`	«Enable operation» with a new target	stays in Operation Enabled
`PVM_ActivateQuickStop()`	«Quick stop»	Quick Stop Active - decelerates on the quick stop ramp, power stage stays on
`PVM_ReleaseQuickStop()`	«Enable operation»	Operation Enabled
`DisablePower()`	«Disable voltage»	Switch On Disabled - power stage off, from any state, shaft coasts
`ClearFault()`	«Fault reset»	Switch On Disabled, if the cause is gone
`GetDeviceState()`	reads the statusword	the current state; `DeviceStateName()` gives its name

Every method that changes the state waits until the device reports the new state and throws if it does not - the message names the state that was expected and the one the device is in. The device enters **Fault** on its own when it detects an error (supply, overcurrent, sensor, ...); the example checks for it in every loop pass and stops.

When something goes wrong

Message	Cause	What to do
`Connect: connection to controller failed (error code: 0x20000004: Opening port error)`	No device on `USB0`, or a different port	Replace the `Connect()` call in `main.cpp` with `escon2.ConnectUsingGUI()` once; the dialog lists the values valid on this PC. Then put them into `Connect()`.
`Connect: device handle is 0 - device already in use?`	Another program holds the port	Close Motion Studio or the other instance of the example
`ESCON2 entered state 'Fault'`	The device detected an error	Read the error in Motion Studio, fix its cause, restart the example - it clears the fault on startup
`Timeout while switching to 'Ready To Switch On' - device is in 'Switch On Disabled'`	Usually no motor supply, or the enable input is not set	Check the supply and the digital inputs configured in Motion Studio
`PVM_SetParameters ... (error code: 0x06090030: ...)`	A value is out of range	Lower `MAX_VELOCITY` below *Max motor speed*
`MotionCmd64.dll was not found` at start	The DLL is not next to the `.exe`	Run the executable from the build folder, or copy the DLL next to it

Error messages from the Motion Library carry the abort code and its description. The codes are listed in *Motion Library Specification.pdf* in the `Manuals` folder of the Motion Library installation.

Adapting the example

- **Connection**: the four strings passed to `Connect()` at the top of `main()`. `ConnectUsingGUI()` shows the valid values.
- **Velocity limit and ramps**: `MAX_VELOCITY` and the four values passed to `PVM_SetParameters()`.
- **Other objects**: `ReadObject()` / `WriteObject()` with the constants in `source/MotionObjects.h`; declare further objects there.
- **Another operation mode**: derive a class from `MotionDeviceBase` like `MotionDevicePVM`, add the mode to `OperationMode` and its objects to `MotionObjects.h`, and register the new `.cpp` and `.h` in `motion_helper.cmake`.