REMOTE UPDATE OF ENOVATION MPC20 CONTROLLERS USING SENQUIP DEVICES

1. Introduction

The Enovation PowerCore MPC-20 is a rugged industrial engine controller used in pumping, power generation and other engine-driven applications. It provides engine monitoring, protection, automatic and manual starting, and configurable control functions.



Figure 1 - PowerCore MPC-20

MPC-20 controllers can be programmed using configuration files generated by PowerVision Configuration Studio software.

Senquip devices provide a method for transferring these configuration files to an MPC-20 over CAN bus. This allows configuration updates to be performed remotely, without requiring a technician to connect a programming computer directly to the controller.

The capability was developed by Senquip in collaboration with Macquarrie Corporation, the Australian and New Zealand distributor of Enovation and Senquip products for industrial and mobile equipment.

2. Supported Senquip Devices

Remote MPC-20 programming is supported on the following Senquip devices:

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

- **QUAD-C2**
- **QUAD-C3**
- **ORB-C3**
- **UNI-Z1**

The example in this Application Note was successfully tested using a Senquip UNI-Z1-S connected to an MPC-20 via CAN bus.

Programming is initiated using the Senquip scripting function SQ.remote_update().

3. How It Works

The programming process consists of five steps:

1. Prepare a suitable MPC-20 configuration file (.gciBin).
2. Upload the configuration file to the Senquip device.
3. Activate a Portal Trigger Parameter to initiate programming.
4. Monitor the progress and error messages reported by the Senquip device.
5. Confirm that the MPC-20 has been successfully updated.

The configuration file is stored locally on the Senquip device before the programming process begins.

Important: Ensure the connected machinery is in a safe state before programming. Updating an engine controller can interrupt its normal operation. Equipment should be secured against unintended starting or operation, and an appropriate recovery procedure should be available.

4. Hardware Connections

Connect the Senquip CAN interface to the MPC-20 CAN bus.

For this example, CAN1 on the UNI-Z1-S was used.

Ensure that:

- **CAN High and CAN Low are correctly connected.**
- **Both devices are powered.**
- **CAN bus termination is correct.**
- **The CAN communication settings are compatible with the MPC-20.**
- **The MPC-20 is not controlling equipment that could operate unexpectedly during programming.**

The procedure can use another available CAN interface if the script and configuration are updated accordingly.

5. Senquip CAN Configuration

The chosen CAN peripheral on the Senquip device must be turned on, and transmit must be enabled. The following settings were used during testing on the UNI-Z1-S:

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

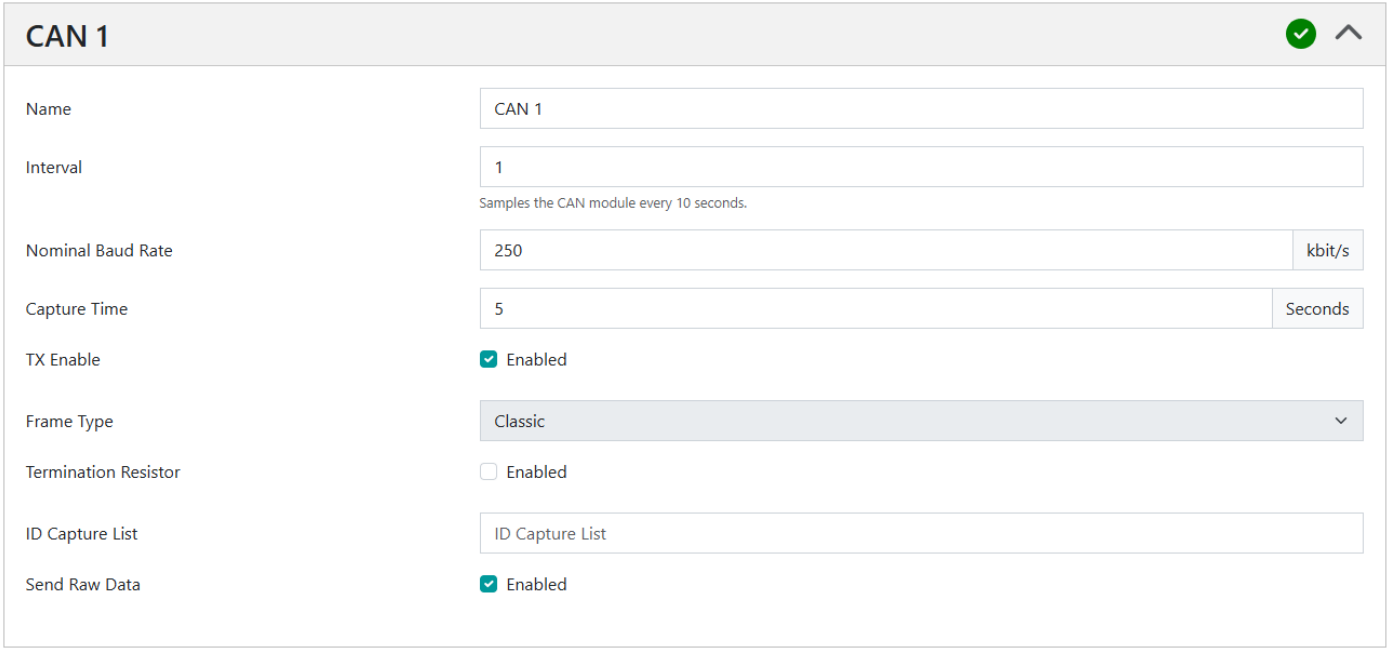


Figure 2 - Senquip UNI CAN Configuration

Other settings, including Wi-Fi and cellular communications, are installation-specific.

6. Upload the MPC-20 Configuration File

Prepare a suitable .gciBin configuration file for the MPC-20.

In this example, the file is named:

Example.gciBin

Using the Senquip Portal:

1. Open the required Senquip device.
2. Navigate to the device's file management functions.
3. Select *File Download.*
4. Transfer Example.gciBin to the Senquip device.
5. Confirm that the file has been successfully transferred.

Files transferred using File Download are stored in the /mnt partition of the Senquip device.

The complete file path for this example is:

/mnt/Example.gciBin

This path must be used when initiating the programming process from within the script.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

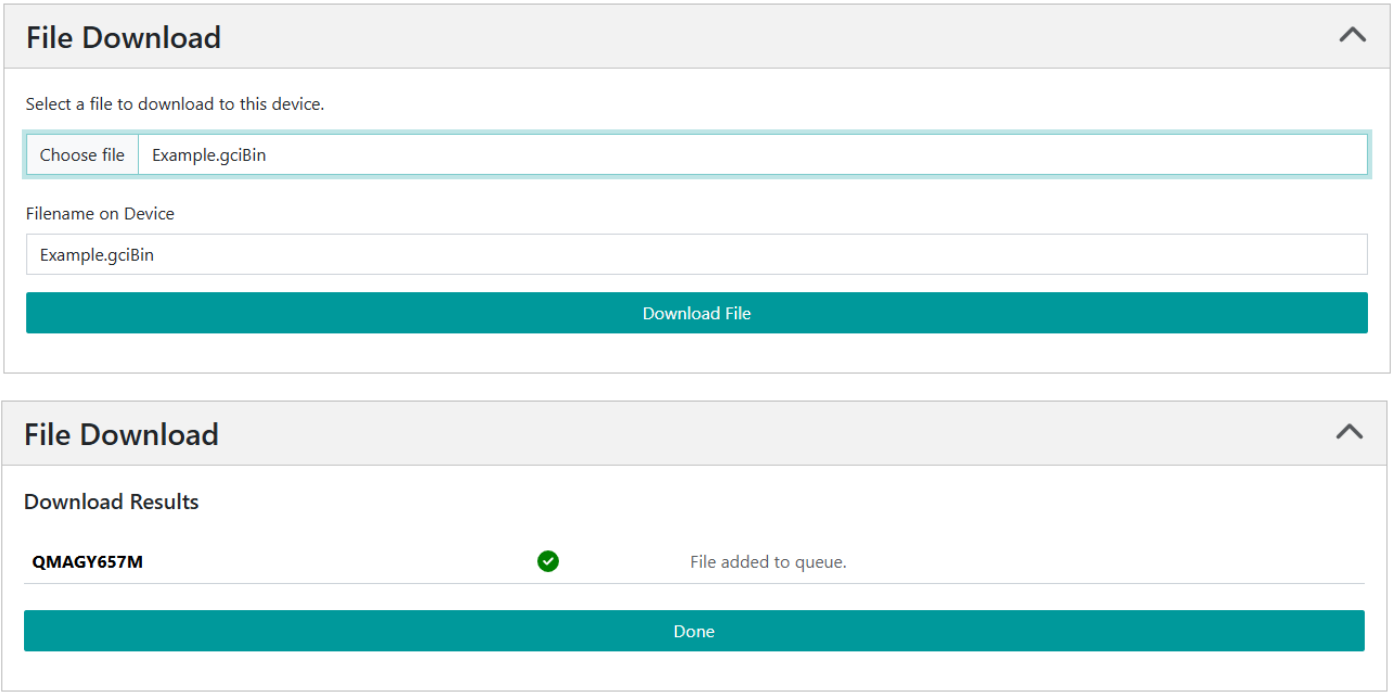


Figure 3 - Downloading Example.cgiBin to the Senquip UNI

You can check that the file is uploaded by listing the files on the device.

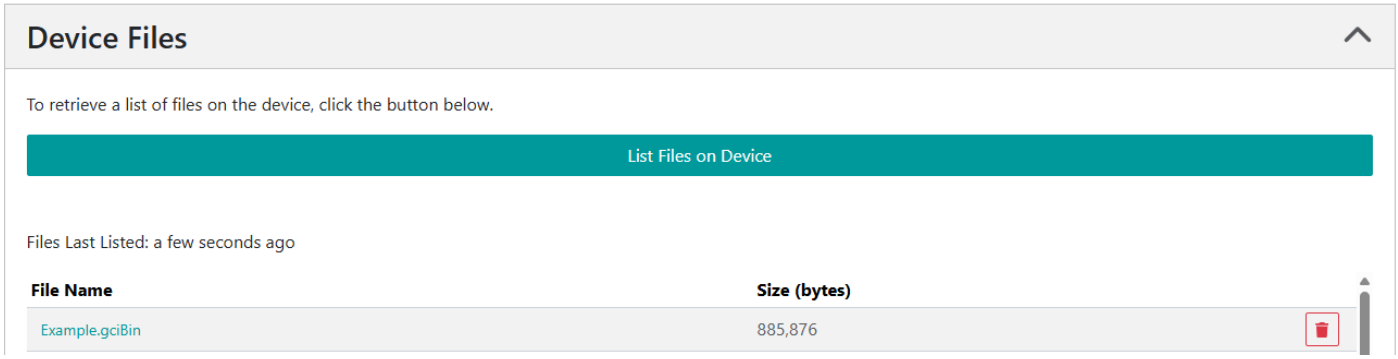


Figure 4 - List of Files on the Senquip UNI

Important: Example.gciBin is an example filename. The file contents must be appropriate for the target MPC-20. The filename used in the script must exactly match the uploaded file, including capitalisation.

7. Senquip Device Script

The following script initiates MPC-20 programming when *Trigger Parameter 1 (TP1)* is activated.

Programming progress and error information are reported using *Custom Parameter 1 (CP1)*.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

```
load('senquip.js');

let last_status = '';

SQ.set_data_handler(function() {
  SQ.dispatch(1, last_status);
  last_status = '';
}, null);

SQ.set_trigger_handler(function(tp) {
  if (tp === 1) {
    let error = SQ.remote_update("/mnt/Example.gciBin", "can1", function(code, percent, msg) {
      last_status = JSON.stringify({code: code, percent: percent, msg: msg});
    }, null);
    if (error !== 0) {
      last_status = JSON.stringify({code: error, percent: 0, msg: 'Failed to start'});
    }
  }
}, null);
```

When TP1 is activated, the script calls:

SQ.remote_update("/mnt/Example.gciBin", "can1", callback, null)

The function attempts to initiate programming using the specified configuration file and CAN interface.

During programming, the callback provides three values:

Parameter	Description
code	Programming status or error code
percent	Reported programming progress
msg	Status or error message

These values are formatted as a JSON string and dispatched through CP1.

If SQ.remote_update() returns an immediate error, the script reports the error code and the message Failed to start. For instance, if the file to be downloaded is not found on the device, the following error is reported.

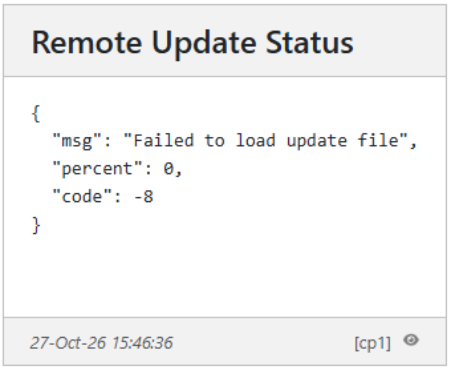


Figure 5 - Error Message when File to be Downloaded is not Found

The script clears last_status after dispatching it. Consequently, CP1 reports the latest available update information rather than repeating the same status indefinitely.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

7.1. Changing the Configuration File or CAN Interface

The filename and CAN interface can be changed by modifying the function call.

For example, when using CAN2:

SQ.remote_update("/mnt/Example.gciBin", "can2", callback, null);

The selected interface must be configured correctly before programming begins.

8. Initiating the Update

Once the configuration file and script have been loaded:

1. Confirm that the MPC-20 is powered and connected to the Senquip device.
2. Activate TP1 to begin programming.
3. Monitor CP1 for programming progress and errors.

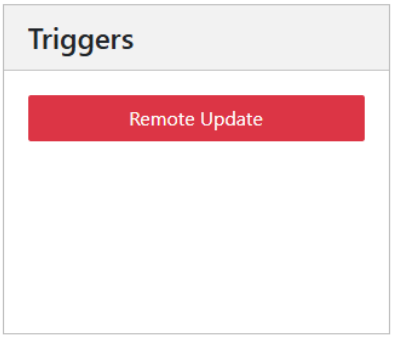


Figure 6 - Use a Trigger to Initiate Download

8.1. Trigger Expiry

The Senquip Portal supports expiry times for Trigger Parameters.

For remote programming, we recommend configuring TP1 with an appropriate expiry period.

This prevents a programming command from remaining queued indefinitely if the Senquip device is offline.

Existing triggers default to Queue until the device connects, preserving previous behaviour. An expiry period must be explicitly configured to prevent delayed delivery.

Trigger expiry and cancellation only apply while a command is awaiting delivery. They cannot stop or reverse programming once the command has been delivered and the update has started.

8.2. Monitoring Programming Progress

The programming process may take some time, depending on the configuration file and communication conditions.

The script reports progress using CP1.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

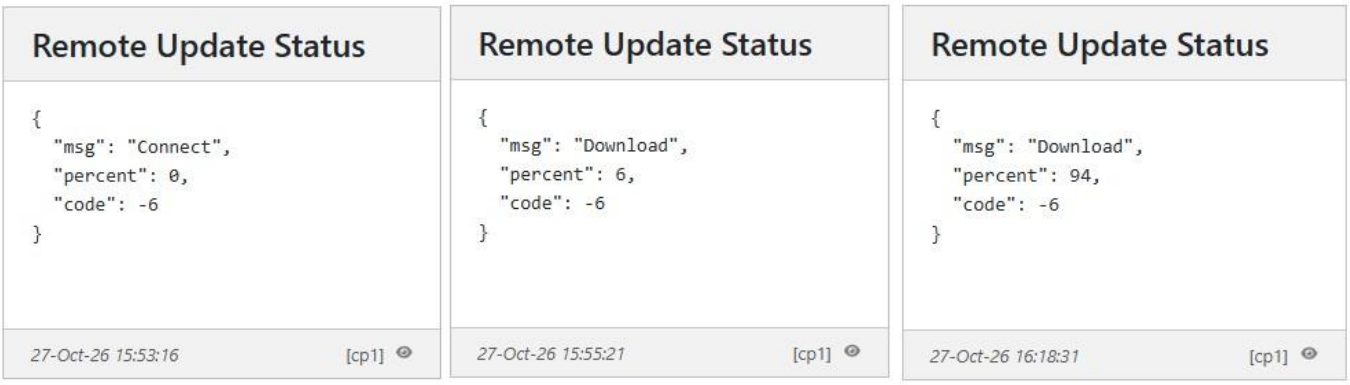


Figure 7 - Download Progress

If an error occurs, use the reported code and message to help identify the cause.

The MPC-20 also provides a visual indication during programming with the yellow light flashing periodically.



Figure 8 - MPC20 During Update

Avoid interrupting power or CAN communications while programming is in progress.

Do not initiate another update until the current programming operation has completed.

9. Successful Completion

When programming completes successfully:

- The uploaded configuration file is deleted from the Senquip device filesystem.
- The Senquip device automatically restarts approximately 10 seconds after successful completion.

Allow the Senquip device to restart and reconnect before continuing.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.



Figure 9 - MPC-20 Successfully Updated

Once programming has finished, verify that the MPC-20 is operating with the intended configuration.

Before returning equipment to service, check the required controller settings, engine protection functions, inputs, outputs and application-specific operating behaviour.

A completed file transfer or programming progress indication should not be considered a substitute for verifying correct controller operation.

10. Conclusion

Senquip devices provide a practical solution for remotely programming Murphy MPC-20 controllers over CAN bus, reducing the need for technicians to travel to site to perform configuration updates.

By combining remote file transfer, programmable scripting and progress reporting, MPC-20 configurations can be updated on equipment deployed in the field. This capability is particularly valuable for remote pumping stations, generators and other engine-driven equipment where site access is difficult or costly.

For more information on integrating remote MPC-20 programming into your application, please contact Macquarrie Corporation or your local Senquip distributor.

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

Appendix A – Full Script

```
load('senquip.js');

let last_status = '';

SQ.set_data_handler(function() {
  SQ.dispatch(1, last_status);
  last_status = '';
}, null);

SQ.set_trigger_handler(function(tp) {
  if (tp === 1) {
    let error = SQ.remote_update("/mnt/Example.gciBin", "can1", function(code, percent, msg) {
      last_status = JSON.stringify({code: code, percent: percent, msg: msg});
    }, null);
    if (error !== 0) {
      last_status = JSON.stringify({code: error, percent: 0, msg: 'Failed to start'});
    }
  }
}, null);
```

Copyright © 2026 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.