ENGINE CONTROLLER HIGH SPEED OVERSIGHT UNIT

By WM International Engineering, Darien, IL - USA

User Manual

Version 4.1

2021-January

Contents

1. Overview

ECHO is an **E**ngine **C**ontroller **H**igh speed **O**versight unit designed to provide controllers such as engine ECUs with the ability to incorporate diagnostics and real-time feedback control on critical systems.

ECHO has embedded software that can run on its own along with ECU and PC software that allows the user to configure and calibrate the unit. It has data acquisition channels targeting critical and high-speed events. Data can be sampled at rates up to 200kHz on each channel, and real-time analysis is performed with a powerful 1.5 GHz processor. The analysis result is sent to ECU via industry-standard CAN protocol and to PC software via ethernet for visualization and calibration.

With ECHO's platform, both hardware and software can be tailored to your application needs. It can conveniently be incorporated into your current workflow for dedicated engine modeling and analysis.

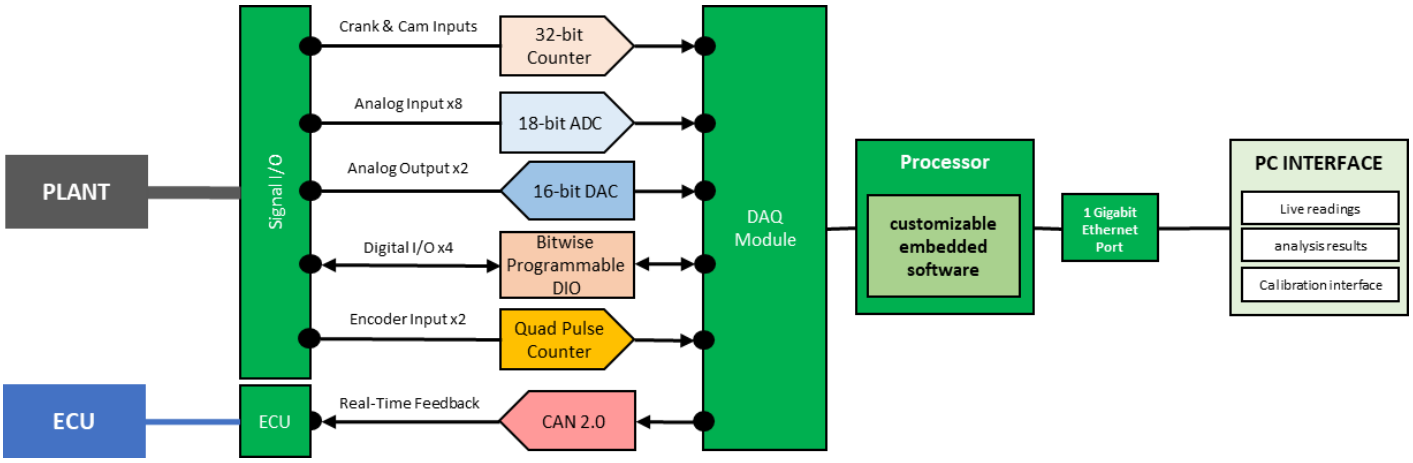


Figure 1: ECHO functional block diagram.

2. Specifications

Each ECHO unit has three connector interfaces, as shown in Figure 2. The main connector goes with a harness that connects to all the digital and analog pins. The ethernet port connects to PC software for live data viewing and calibration. The UCB-C port provides the power to the ECHO.

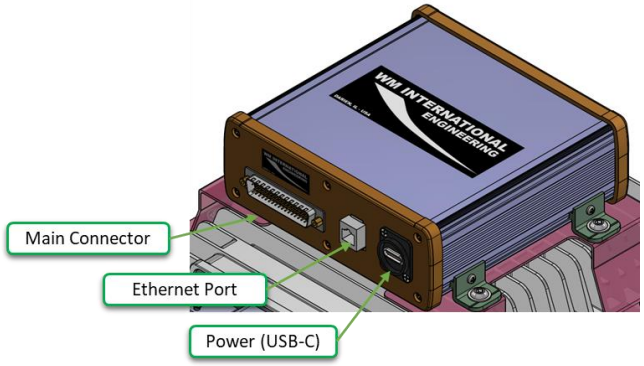


Figure 2: ECHO connection interface.

The connection specification is shown in Figure 3 and the pinout is shown in Figure 4. There is a pair of Crank and Cam inputs, eight analogs inputs, two analog outputs, and four digital I/O. The digital I/O can be configured to be either as input or output and is used as E-Stop by fault. The encoder inputs are not listed but available if needed.

The connectors for the harness endpoint can be tailored to the application's needs. Note that the Crank and Cam digital inputs can accommodate both hall effect and variable reluctance sensors. Please contact our team for customizing the hardware to your application.

Power Connection	**Nominal**	**max**	**min**	**Connector Type**
Power Supply	5V DC	5.25	4.5	USB-C
Power Consumption	10W	15	0.9225	
DATA ACQUISITION				
DIGITAL				
Cam	5V	2.2 .. 30V	-0.5 .. 1.5V	
Crank	5V	2.2 .. 30V	-0.5 .. 1.5V	
ANALOG INPUTS 0,1,2,3,4,5,6,7	-10 .. 10V	15	-15	
Sampling resolution	50kHz / 0.5deg	200kHz / 0.1deg		
Bit Resolution	18-bit			
ANALOG OUTPUTS 0,1				
DIGITAL I/O 0,1,2,3	50kHz / 0.5deg	200kHz / 0.1deg		
Bit Resolution	18-bit			
CAN Bus				
Connector 2,3,7	CAN Low, GND, CAN High			9-pin D-SUB Male
Ethernet Connection				
Suggested Adapter				USB 3.0
PROCESSOR				
CPU	Quad Core Cortex-A72; 64-bit SoC @1.5GHz			
RAM	2GB LPDDR4; 2400 SDRAM			
STORAGE	16GB			
OS	Linux Based			

Figure 3: Connection specifications.

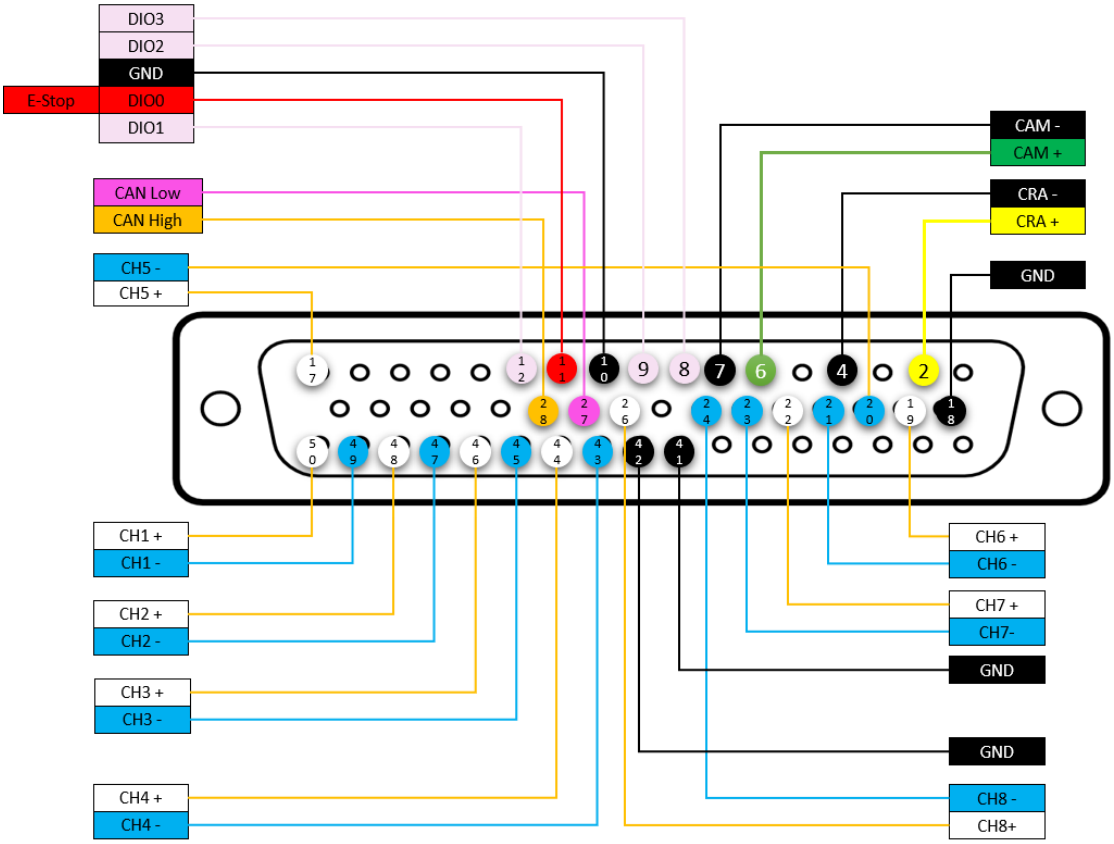


Figure 4: Connector pinout for cylinder pressure feedback application.

3. Software Installation

ECHO comes with license-free graphical user interface (GUI) software and currently supports Windows 10 machines. The standard installation package is downloadable from the product page. However, the software can be customized based on your application. In that case, please request the latest installation package from our team. The installation package contains two files: “Echo Certificate” and “Echo App Installer”.

3.1 Echo Certificate Installation

Certificate installation is required if it is the first time ECHO software is being installed. Otherwise please proceed to 3.2 Echo App Installation.

To install the certificate, open the “Echo Certificate” file to begin and follow the steps: 1) Click on Install Certificate. 2) Select Local Machine. 3) Click next. 4) Click Browse and choose the path “Trusted Root Certification Authorities”. 5) Click next. 6) Click Finish to complete the certificate installation. Please reference Figure 5 for a visual guide.

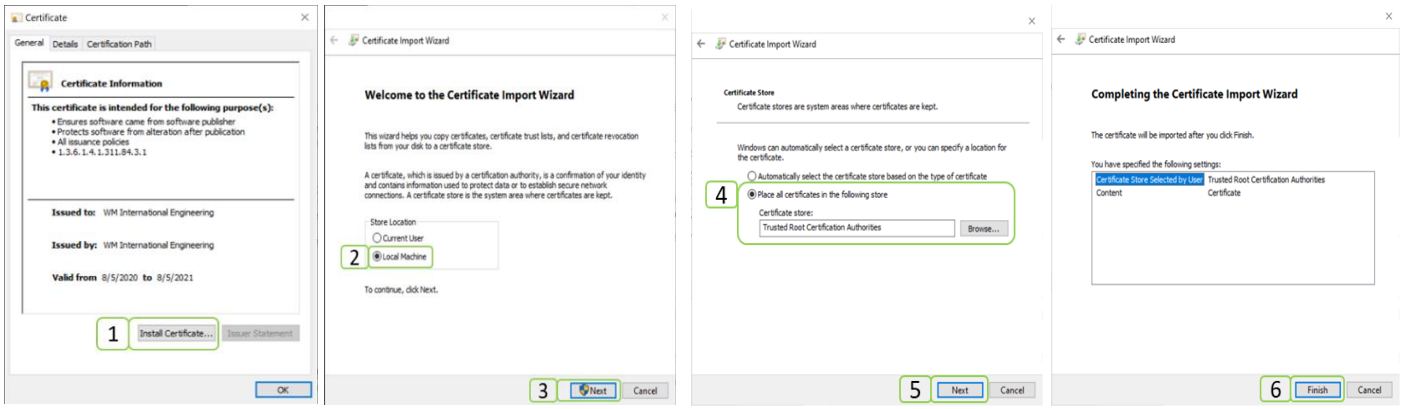


Figure 5: Certificate installation prompt windows and steps.

3.2 Echo App Installation

To begin the process, open the “Echo App Installer” file and then click Install. ECHO software will launch when the installation is complete.

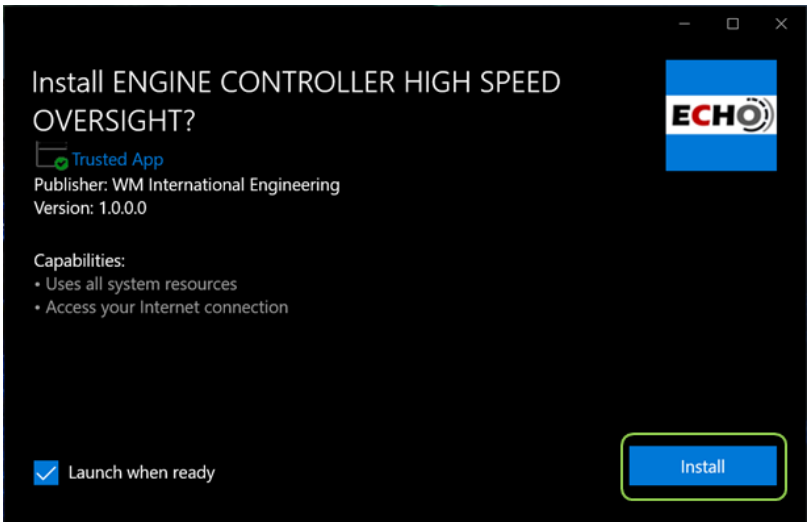


Figure 6: ECHO installer windows.

4. Software Usage

4.1 Banner Control Strip

The banner control strip in Figure 7 consists of the following:

- Setup button – go to the Setup page.
- Measure button – go to the Measure page.
- Analyze – go to the data playback page.
- Diagnostics button – bring up diagnostics window.
- Info button– bring up the information window.
- Status box - list major events and system messages with time stamps.

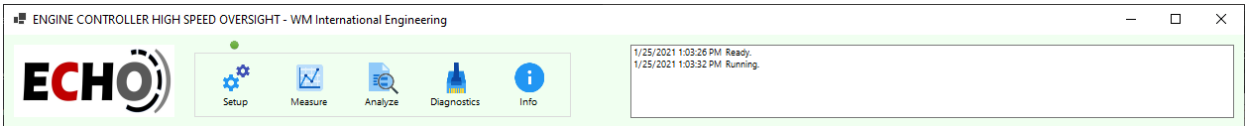


Figure 7: Banner control buttons and the status message box.

4.2 Setup Page

Click on  will bring up the setup page. It allows the user to perform a software update and update engine and channel configurations, as shown in Figure 8.

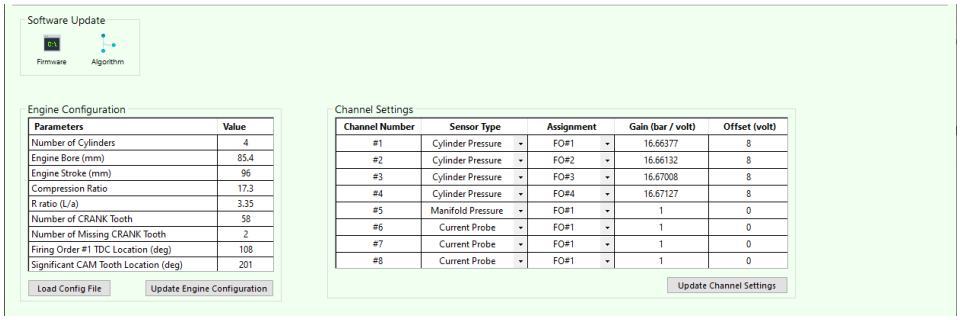


Figure 8: Setup page controls - Software Update, Engine Configuration, and Channel Settings.

Software Update:

- Click  to update the firmware version on ECHO. This process overrides its embedded software, including analysis algorithms and calibrations.
- Click  to upload a customized combustion analysis algorithm “Analysis.c”. This affects how ECHO does its combustion analysis. Please contact our team to request the latest source code.

Engine Configuration:

- Use the table to adjust engine parameters, which are related to sync logic and combustion analysis.
- The “Load Config File” button allows the user to upload a pre-made file that automatically populates both engine configuration and channel settings tables.
- Use the “Update Engine Configuration” button to send the updated engine configuration to ECHO.
- For entry explanation, please see section *Engine Configuration Details* for more.

Channel Settings:

- Use the table to assign channel settings and gains.
- The “Update Channel Settings” button will update the channel setting and restart ECHO.

4.3 Measure Page

The measure page allows the user to connect to ECHO, view/record data, and perform live calibration. As shown in Figure 9, there is a control strip, data file setting, analysis result table, plot input, calibration table, and plot area.

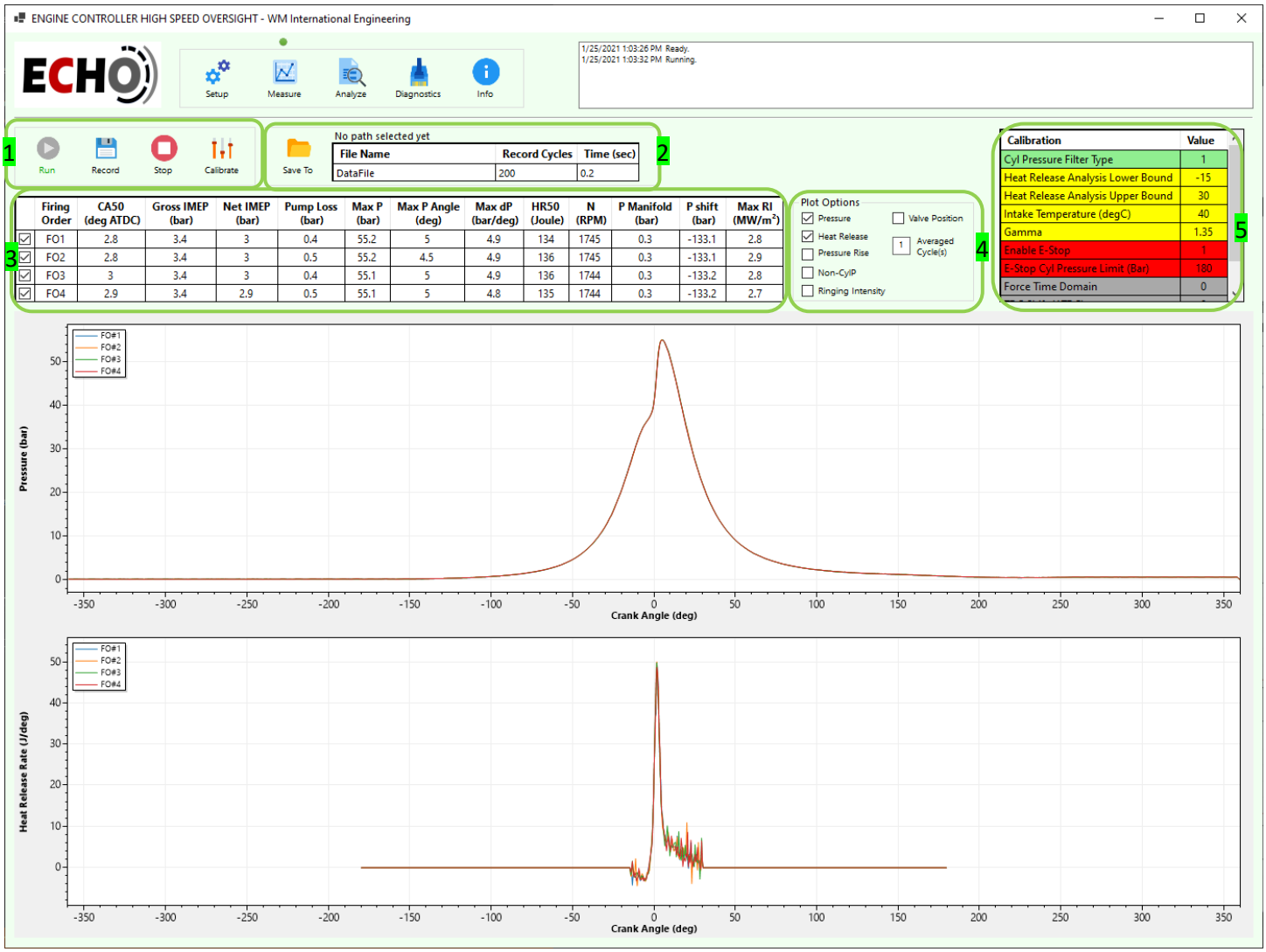


Figure 9: Measure page showing (1) control strip, (2) data file setting, (3) analysis result, (4) plot options, (5) calibration table, and plot area.

(1) The control strip has the following functions:

- Click to start data transmission. The application will automatically perform this action if the device is detected.
- Click to record data. Recording status will appear in the status box.
- Click to stop data transmission. It pauses live data transmission. This is only required when performing firmware or algorithm updates.
- Click to show or hide the calibration table. The table contains calibratable parameters when running.

(2) The data file setting table has the following functions:

- Click to select a folder where the recorded data will be store.
- Use the table to change the file name, the number of cycles to record, and the duration to record. The date and time will automatically be appended to the filename when saving.
- If ECHO is in sync with the engine and analysis data is available, record cycles will be used. The recorded data will be in the crank angle format. Otherwise, recorded data will be in the time domain format, and the inputted time duration will be used.

(3) The analysis result table and plots are automatically populated when data is available. Check or uncheck the firing order to show or hide the according plot line.

(4) Use the plot options to view desired plots. The number of averaged cycles can be adjusted for visual purposes. After data for the specified cycles has been received, the result table and plots will be updated. The plot area is interactive. Please right click on any of the plot and click “help” to see more plot viewing tips.

(5) The calibration table shows live adjustable parameters. While running, any changed valve will be used immediately. Table 1 below contains descriptions for each item.

Table 1: Live calibratable parameters.

Variable	Description
Cyl Pressure Filter Type	The integer value indicates the type of filtering applied to the in-cylinder pressure at the start of combustion analysis (0 = No filter, 1 = three-point moving average filter, 2 = five-point moving average filter).
Heat Release Analysis Bounds (Upper and Lower)	The bounds used for heat release rate computations. Only the pressures in this window will be considered when performing heat release analysis.
Gamma	The ratio used in combustion analysis.
Enable E-Stop	Boolean value for arming E-Stop (0 = enable, 1 = disable).
E-Stop Cyl Pressure limit	Cylinder pressure limit trigger limit for E-Stop.
Force Time Domain	Boolean value input for operation mode (0 = crank angular domain, 1 = time domain). Note that no combustion analysis will be performed if it is in the time domain.
TDC Shift (ATDC)	Degrees to shift the crank angle relative to Firing Order #1 Location (set engine configuration). Positive angles for shifting to right (delaying). Negative angles for shifting to left (advancing). The resolution is 0.1°.
Unique CAM Edge Shift (ATDC)	Degrees to shift the window to look for the unique CAM Edge relative to Significant CAM Tooth Location.

Time-domain Mode

ECHO will go into time-domain mode if it is not synced with the engine or forced to operate in the time domain (via calibration window). It is intended for troubleshooting sync or sensor reading issues, as all the raw data is sent to the PC. A trigger feature is provided for the convenience of identifying the timing among the analog channels and crank and cam signals. Figure 10 shows an example of the interface in the time domain.

Figure 10: Measure page in time-domain mode - (1) statistics table and plots of raw data beneath (2) trigger setting, (3) duration input.

The duration of the plot is based on the input in the data file setting. The trigger position is at the mid-point of the duration. For instance, if the total is duration is 0.2 seconds, the trigger will happen at 0.1 seconds. The statistics table will display voltage at the trigger, maximum, minimum, and average voltages over the sweep. Right click on the plot area to hide or show desired channels.

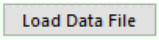
There are four adjustable trigger parameters: 1) trigger mode, 2) trigger edge, 3) trigger channel and 4) trigger level. Please see Table 2 for details for each parameter. The recorded data will be at the highest available resolution (i.e. 50kHz or 200kHz).

Table 2: Time-domain mode trigger parameters.

Parameters	Description
Trigger Mode	Determine how the sweep happens. In rolling mode, data will be plotted as it comes in and roll from right to left of the plot. In normal mode, the shown data will only sweep if the input signal reaches the set trigger level. In auto mode, a timer will trigger the sweep even without reaching the trigger level. In single mode, the data will only sweep once.
Trigger Edge	Select the edge type. The trigger can be done on either rising or falling edge.
Trigger Channel	Select the trigger source. Available trigger channels are analogs #1 through #8.
Trigger Level	Determine the trigger level. The range is in voltage from -10V to +10V.

4.4 Analyze Page

The analyze page provides an interface for the user to load a data file and playback the data cycle by cycle. While testing the engine, the user can use this feature to swiftly review the data and change relevant calibrations based on the area of interest.

Figure 11 is an example of the analyze page. Use 

button to load the desired data file. The selected file source path will display next to the button, and the file information will be in the info table beneath. Click the play button to begin the playback. The data will be displayed cycle by cycle, and the current position is shown in the file info table. The progress bar next to the play button will also match the current position. If the playback is paused, the pointer on the progress bar can be moved to index to a specific cycle. Click on the left or right side of the pointer on the progress bar will index the cycle backward or forward by one. Please refer to the software demo on the product page for more visual guidance.

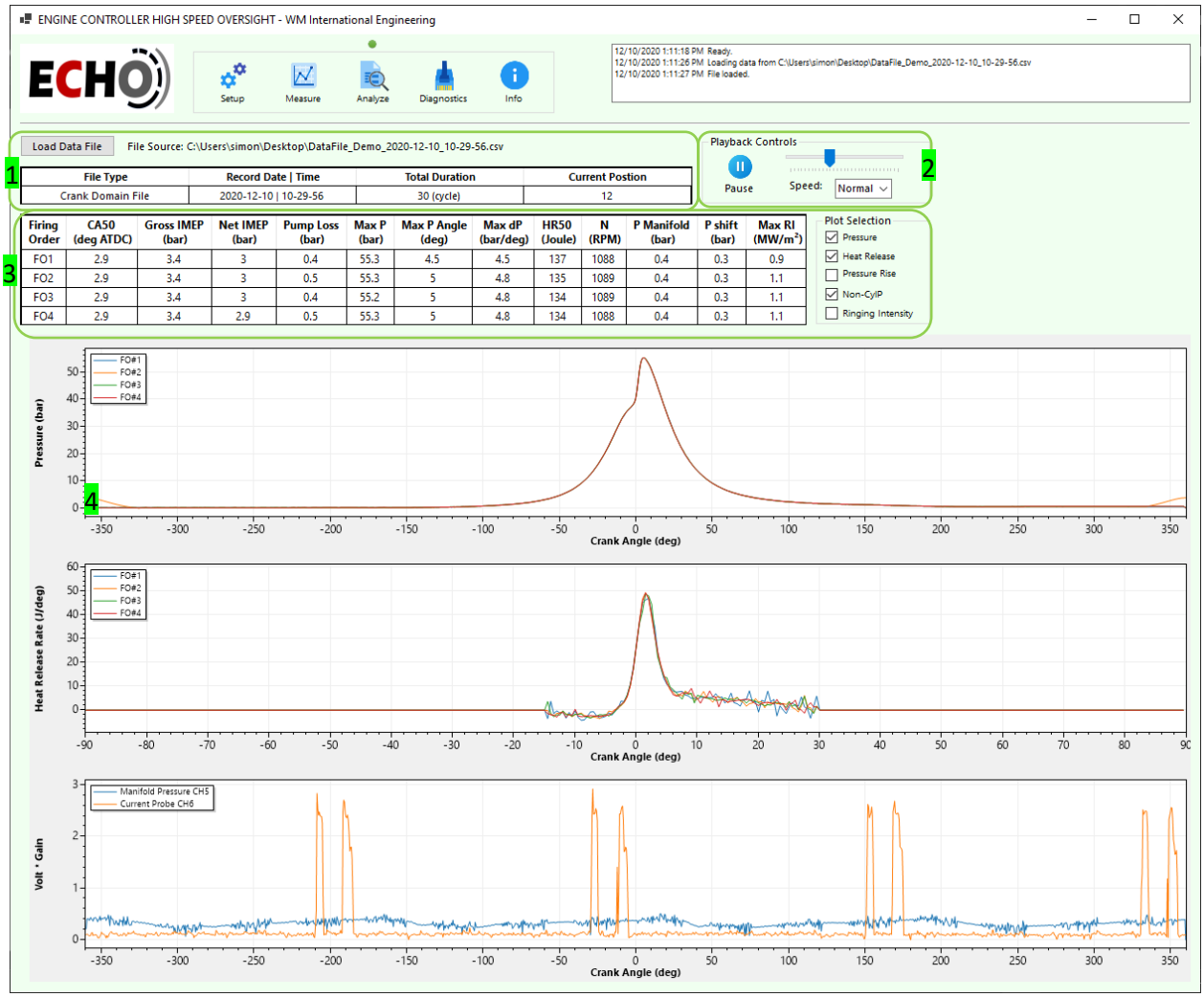


Figure 11: Analyze page showing (1) file into panel, (2) play back control, (3) scalar data table, (4) plots of the data arrays.

4.5 Diagnostics Window

Diagnostics are useful when having issues with connection and engine synchronization. Click  on the banner control to bring up the diagnostics window shown in Figure 12. The details for each item are described in Table 3. There are also cycle based running plot of the analysis table. This can be used to identify the variation among the cylinders.

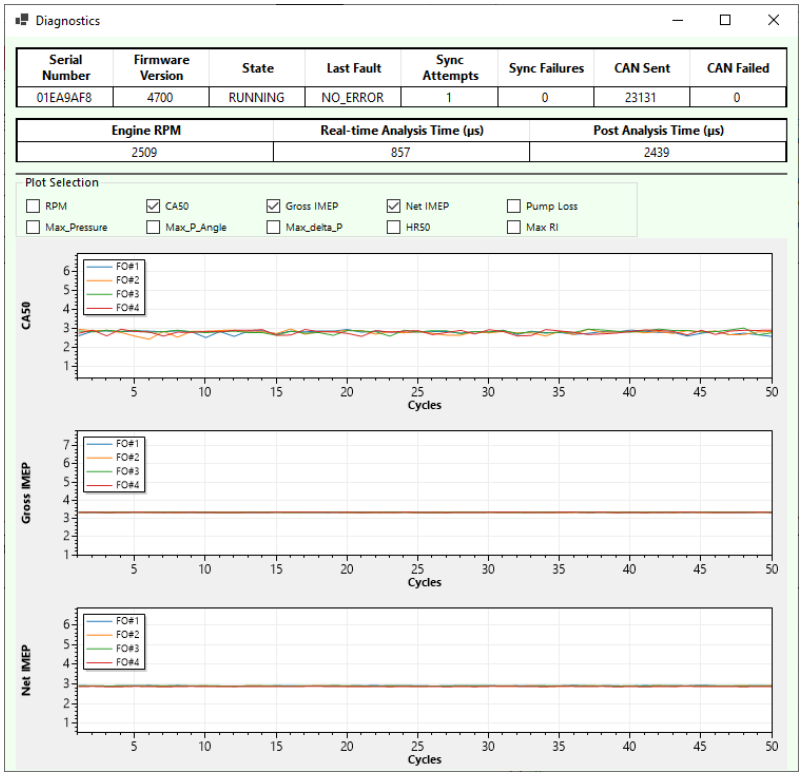


Figure 12: Diagnostics window.

Table 3: Diagnostics window item entry description.

Variable	Description
Serial Number	Serial number of the connected device.
Firmware	Firmware version on the connected device.
State	Current running status of the device.
Last Fault	Last error encountered.
Sync Attempts	Number of times attempted to synchronize with the engine.
Sync Failures	Number of times failed to synchronize with the engine
CAN Send	Number of CAN messages sent into the CAN bus.
CAN Failed	Number of CAN messages failed sending.
Engine RPM	Engine's revolution per minute
Realtime Analysis Time (µs)	Time taken to perform real-time analysis (-180° to +180° ATDC)
Post Analysis Time (us)	Time taken to perform post-analysis and send data to PC (-360° to +360° ATDC)

4.6 Info Window

Click to bring up the info window, which contains some quick reference information about ECHO, such as CAN message format and connector pinout. Figure 13 is an example of the window showing the connector pinout.

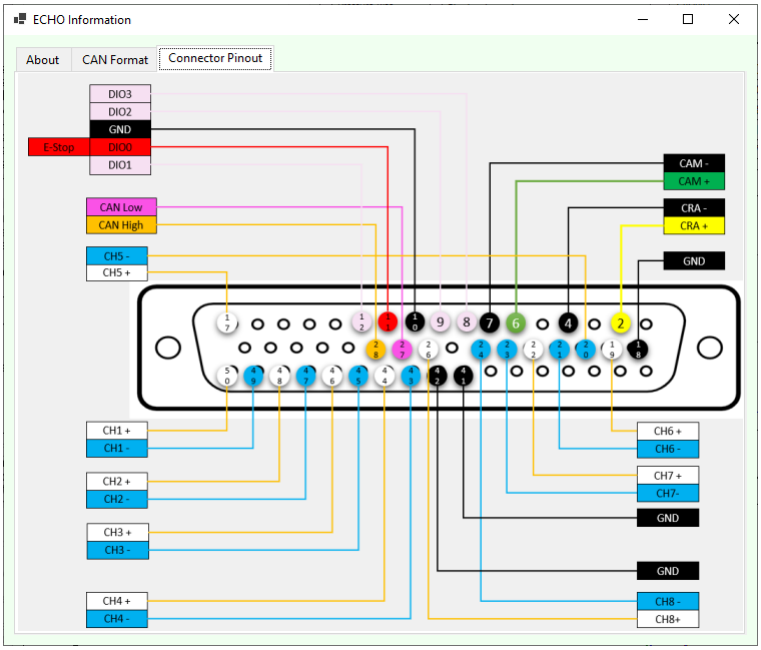


Figure 13: Info window showing connector pinout.

5. CAN Interface

ECHO send ECU the latest analysis result, such as CA50 and max pressure rise rate, via CAN bus. The convention used is the little-endian (intel) format, which means that lower bytes are written first. Values are scaled up to unsigned integers to improve transmission efficiency. When receiving on the ECU side, a scale factor needs to be applied. Details are in Table 4.

Table 4: Outgoing CAN Message content and format.

Variable	Description	Start Bit	Bytes	Scale Factor for ECU
CAN Message ID (Firing Order)	standard CAN ID 0x100 = FO1 0x101 = FO2 0x102 = FO3 0x103 = FO4	N/A	2	N/A
PRR	Scaled double of max pressure rise rate	0	2	x / 10
CA50	Scaled double of CA50	16	2	(x - 180) / 10
Max P	Scaled double of max pressure	32	2	x / 10
IMEP	Scaled double of gross IMEP	48	2	(x - 180) / 10

On the same CAN interface, ECHO is capable of receiving information from ECU and re-direct it to the PC software for visualization. For example, ECHO can get the valving duration and timing data from the ECU and overlay it with cylinder pressure and piston position. As shown in Figure 14, the effect of the valve control on the cylinder pressure can be observed live.

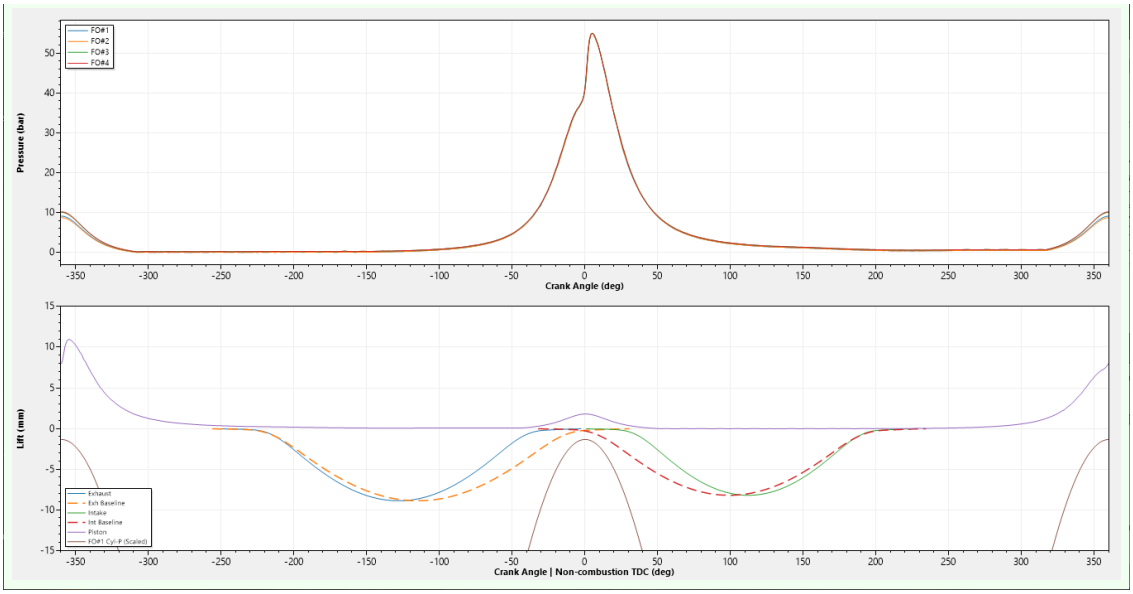


Figure 14: ECHO Valve actuation application, showing live valve position and cylinder pressure relative to piston motion.

The visualization feature requires both code modification on embedded and PC software based on the specific application. Please contact our team with your needs so that the software can be tailored to your strategies.

6. Engine Configuration Details

Engine configuration is used for inputs related to synchronization and combustion analysis. The configuration is stored on ECHO and loaded at the beginning of every power cycle. Table 5 contains an explanation for each entry, and Figure 15 is a visual aid for parameter inputs related to engine synchronization. The parameters can be updated via PC software.

During run time, ECHO will detect the falling edge of the crank and cam pulses. It will sync with the engine by successfully identifying crank and cam teeth have found within an engine cycle. The synchronization happens in two stages. The first stage is to sync with the crank wheel based on the number of regular and missing teeth. Once the crank wheel is synced, it will look for the significant cam pulse to determine the current cycle (intake vs combustion) of the engine. After successfully identifying the cam pulse, ECHO will switch to analysis mode and keep track of the number of the crank teeth found to ensure the synchronization is valid.

ECHO's synchronization strategy is similar to how an engine ECU would sync at cranking. If ECHO is unable to sync with your engine using the available parameters, please send us your crank and cam profile. We can provide a software update to accommodate the profile.

Table 5: Engine configuration entry explanation.

Variable	Description	Purpose
Number of Cylinders	Number of cylinders	Determine analysis and feedback timing
Engine Bore	Bore in millimeters	Derive cylinder volume
Engine Stroke	Stroke in millimeters	Derive cylinder volume
Compression Ratio	Ratio of max and min volumes	Derive cylinder volume
R Ratio	Ratio of connecting rod and crank arm	Derive cylinder volume
Number of Regular Crank Tooth	Number of teeth on crank wheel	Sync with engine, interpolation
Number of Missing Crank Tooth	Number of missing teeth on crank wheel	Sync with engine, interpolation
Firing Order #1 TDC Location	TDC#1 location relative to 1st crank tooth	Calculate crank angle, execution timing
Significant CAM tooth Location	Unique CAM pulse location relative to 1st crank tooth. There should be no additional CAM pulse within ±10° of the unique CAM pulse.	Sync with engine

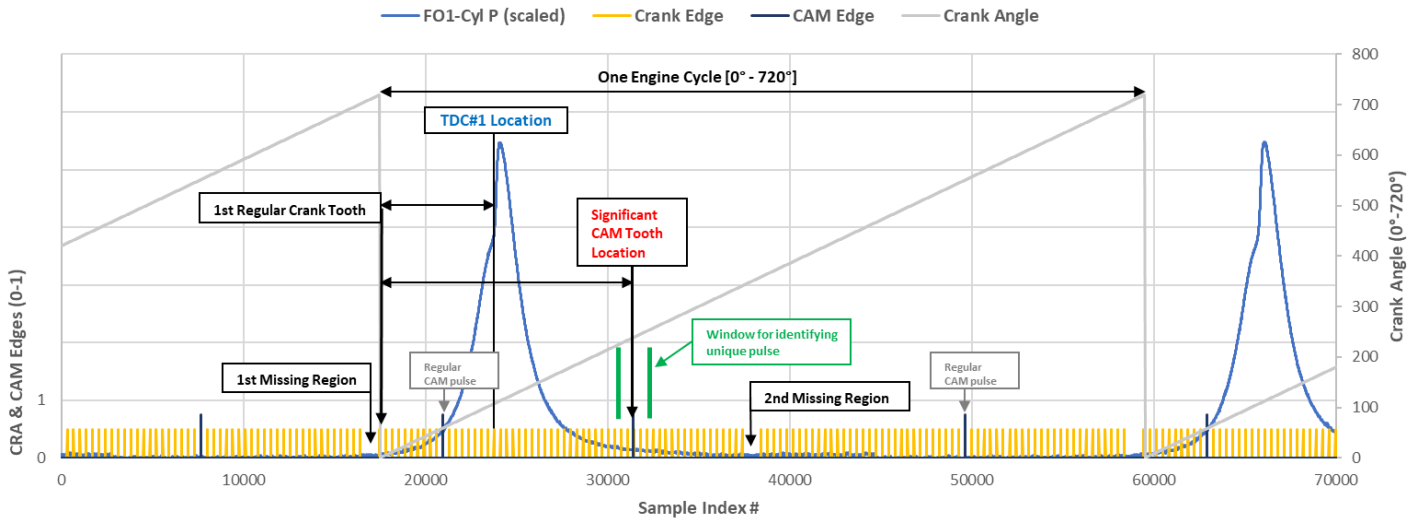


Figure 15: Engine sync visual aid, indicating TDC#1 and significant CAM tooth locations relative to 1st regular crank tooth.

7. Combustion Feedback Speed Specification

ECHO has two variants that come with different sampling frequencies. Please see Table 6 for the specification. The interpolative angular sampling resolution is set programmatically in the embedded software. The default option is at 0.5°. Available resolutions are 0.1°, 0.5°, and 1.0°. Both ECHO models can sample data at any of the available resolutions. If your application requires a different interpolative sampling option, please connect with our team for customization.

Table 6: ECHO sampling specification.

Model	Sampling Frequency	Interpolative Sampling Options	Actual Sampling Resolution in Crank Domain	Maximum Sampling Error
ECHO High Speed	50,000 Samples/Sec	0.1°, 0.5°, 1.0°	0.5° up to 4,100 RPM 1.0° up to 8,200 RPM 2.4° up to 20,000 RPM	0.25° up to 4,100 RPM 0.50° up to 8,200 RPM 1.20° up to 20,000 RPM
ECHO Hyper Speed	200,000 Samples/Sec		0.1° up to 3,300 RPM 0.5° up to 16,500 RPM 0.6° up to 20,000 RPM	0.05° up to 3,300 RPM 0.25° up to 16,500 RPM 0.30° up to 20,000 RPM

Combustion Analysis Closed-Loop Cycle-to-Cycle Control

ECHO can be used as a combustion diagnostics tool as combustion data can be readily displayed in the PC software. The tool however is designed to provide cycle-to-cycle feedback for immediate control of the fuel system and provide additional information to adjust the engine parameters and control. The capability of ECHO to provide real-time control resides in its ability to perform calculations highly effectively on a dedicated processor and provide efficient communication to the ECU via CAN.

ECHO performs calculations of cylinder-specific combustion phasing and torque based on data acquired within a window of crank angle around the firing TDC, from -180° to +180°. This gives ECHO a window of time to calculate and send data to the ECU ahead of the next firing event.

Performing interpolative sampling, combustion analysis, and sending data via CAN 2.0 bus takes an average of 2.4ms and a maximum of 6.0ms. The results are based on 40,000 cycles, using ECHO High Speed 0.5° resolution and CAN baud rate of 1Mbit/sec. Tests performed on a 4-cylinder engine showed that closed-loop cycle-to-cycle control was guaranteed at engine speeds below 8,000 RPM. The estimates shown use the high range time estimates of 6ms, but it should be noted that the average computation and transmission time is on the order of 2.4ms.

Figure 16 is an illustration detailing the piston (BLUE), cylinder pressure (RED), and Heat Release calculation (GREEN) from a Diesel engine. To consecutive cycles are shown, with the crank angle annotated in the x-axes. The figure captures a combustion event at 4500 rpm. Letters A through G are described in the subsequent table. Points A and B are the beginning and end of the data acquisition. This encompasses a total of 360 degrees. ECHO requires at most 1ms to execute the calculations, which established point C. As noted earlier, the CAN communication requires a maximum of 5ms, which places the time to point D. The ECU requires approximately 2ms to perform the feedback calculation and prepare the command for the injection event. The corresponding crank angles for these processes are labeled E and F respectively. Typical early injection events (or spark timing commands) are at 50 degrees before the firing TDC or 670 degrees on the figure x-axes. At 4500 rpm, the combustion feedback total loop is complete at 450 degrees, significantly ahead of the 670-degree limit. Table 5 also presents estimates at 3000 rpm and 8000 rpm. At 8000 rpm, the combustion total loop is complete at 660 degrees, still below the 670-degree limit.

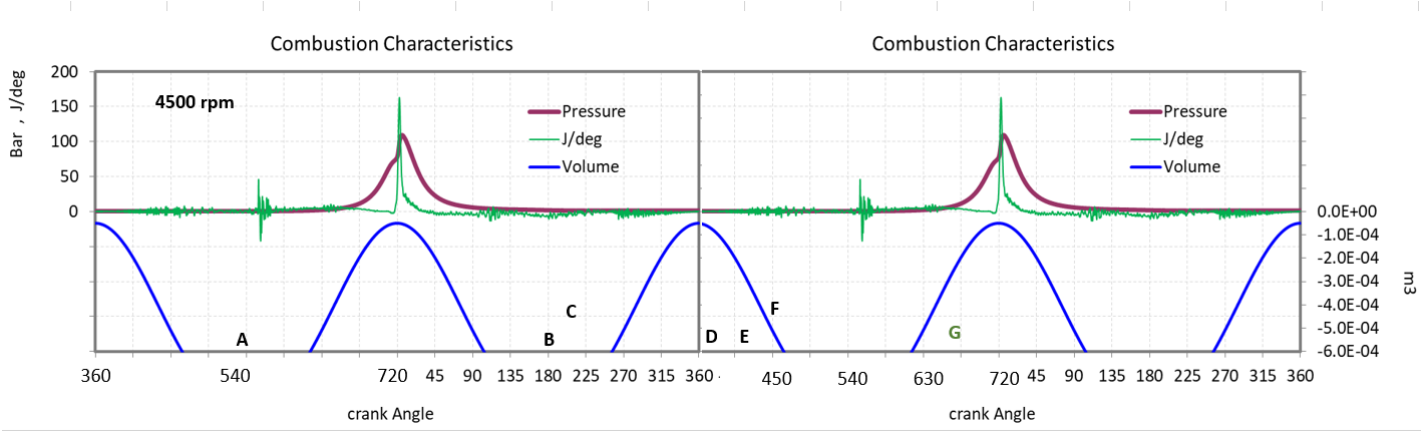


Figure 16: Illustration of critical check points at 4500RPM, corresponding to data in Table 7.

Table 7: Critical check points for close-loop cycle-to-cycle control for 3000RPM, 4500RPM, and 8000 RPM.

Spec	Check Point on plot	RPM			Spec Unit
		3,000	4,500	8,000	
Start of Acquisition	A	540	540	540	deg
Firing TDC		720/0	720/0	720/0	
End of Acquisition	B	180	180	180	
Heat Release Calculation		1	1	1	ms
Heat Release Calculation		18	27	48	deg
Heat Release End	C	198	207	228	deg
CAN Transmission		5	5	5	ms
CAN Transmission		90	135	240	deg
CAN Transmission end	D	288	342	468	deg
ECU Fuel Correction		2	2	2	ms
ECU Fuel Correction		36	54	96	deg
ECU Fuel Correction End	E	324	396	564	deg
ECU Fuel Command		2	2	2	ms
ECU Fuel Command		36	54	96	deg
ECU Fuel Command End	F	360	450	660	deg
Target Start of Ignition (e.g. pilot)	G	670	670	670	deg BTDC

8. Algorithm Update Tutorial

The user has the option to modify the default combustion analysis C source code called “Analysis.c”. The modified analysis algorithm can be uploaded to ECHO via the PC software. This section has two parts: 8.1) a brief walk-through of the default “Analysis.c” file. 8.2) an example to perform the algorithm update.

8.1 Baseline combustion analysis algorithm walk-through

The default combustions analysis algorithm has 5 five functions:

- initializeAnalysis() – this function is called once at the start of the entire program and performs analysis calculations that only need to be done once. For example, the default algorithm used this to calculate cylinder volume array from -360° to + 360° top-dead-center (TDC). Below is a snippet of the entire function.

```
void initializeAnalysis() {
    // Get related calibration values
    double bore = getCal(BORE);     // mm
    double stroke = getCal(STROKE); // mm
    double CR = getCal(COMPRESSION_RATIO);
    double rRatio = getCal(R_RATIO);

    // Calculate displacement and tdc volumes in liters
    double Vdisp = (NUMBER_OF_CYLINDERS * M_PI * stroke * bore * bore / 4.0) * (1000.0 / 1000000000.0);
    double Vtdc = Vdisp / NUMBER_OF_CYLINDERS / 1000 / (CR - 1);

    // Crank angle at start of the array
    double theta = -360.0;

    // Calculate cylinder volume
    // loop through -360° to +360° @ 0.5° resolution
    for (int i = 0; i < 1440; i++) {
        double sinTheta = sin(theta * M_PI / 180);
        double v1 = rRatio + 1 - cos(theta * M_PI / 180) - sqrt(rRatio*rRatio - sinTheta*sinTheta);
        double v2 = 0.5 * (CR - 1);
        volume[i] = Vtdc * (1 + v1*v2); // m^3
        theta += 0.5;
    }
}
```

- analyseData180degATDC() – this function is called for every firing order at 180° deg after TDC and performs time-sensitive calculations needed for cycle-to-cycle feedback control. Analog pressure array from -180° to +180° of TDC is analyzed to find max pressure, max pressure angle, pressure rise rate, heat release rate, gross IMEP, and CA50. Critical parameters, such as CA50 and max pressure rise rate, are sent to engine ECU via CAN. All analysis results are saved to an analysis result structure. Below are highlights of the function.
    - The first major step is to filter the pressure array based on selected filter options: unfiltered, three-point, or five-point filters. The filter option is stored on ECHO locally and can be adjusted via PC side software.

```
// Filter pressure voltage array based on chosen filter
double* inputArray;
double adcPressureFiltered[720];
if (dataFilterType == THREE_POINT_MOVING_AVERGAE) { // 3-point filter
    threePointMovingAveragefilter(adcPressure, adcPressureFiltered, 720);
    inputArray = adcPressureFiltered;
} else if (dataFilterType == FIVE_POINT_MOVING_AVERGAE) { //5-point filter
    fivePointMovingAveragefilter(adcPressure, adcPressureFiltered, 720);
    inputArray = adcPressureFiltered;
} else {
    inputArray = adcPressure; // unfiltered
}
```

- Second, calculate manifold pressure and pressure shift.

```
// Get pressure shift reference pressures
   double sumAdcPressure = 0;
   for (int i = 0; i < 10; i++) {
          sumAdcPressure += inputArray[i];
   }

// Calculate manifold pressure (volt to bar)
   double map = (pManifoldVolts < 0) ? 0 : pManifoldVolts * mapBarsPerVolt;

// Save to analysis result
   analysisResult->pressureManifold = map;
   analysisResult->pressureShift = map - barsPerVolt * (sumAdcPressure/10);
```

- Third, loop through the pressure array, and calculate the pressure rise rate, work, and heat release rate.

```
    // Loop from -180° to 179.5° TDC @ 0.5° resolution
    for (int i = 1; i < 720; i++) {

          // Get current & previous pressure points
        double pressure = inputArray[i] * barsPerVolt + analysisResult->pressureShift;
        double prevPressure = inputArray[i-1] * barsPerVolt + analysisResult->pressureShift;

          // Record max pressure & its angle
        if (pressure > maxPressure) {
            maxPressure = pressure;
            maxPressureAngle = crankAngle;
        }

          // Calculate pressure rise rate
        double pressureRiseRate = (pressure - prevPressure) / deltaAngle;

        // Find max pressure rise rate
        if (pressureRiseRate > maxPressureRiseRate) {
            maxPressureRiseRate = pressureRiseRate;
        }

        // Calculate work (integral PdV)
        integralPdV += pressure * (volumeShifted[i]-volumeShifted[i-1]) * newtonsPerBar;
        if (volumeShifted[i] < minVolume) {
            minVolume = volumeShifted[i];
        }
        if (volumeShifted[i] > maxVolume) {
            maxVolume = volumeShifted[i];
        }

        /* See next snippet...
        *  Record pressure rise rate
        *  Calculate heat release rate
        *  in the significant analysis window (-89.5° to +90° tdc)
        */

        crankAngle += deltaAngle;
    }
```

- While loop through the pressure array, the heat release rate is calculated in specified combustion window limits (i.e. -15° to 30°), which are store locally on ECHO and calibratable via PC software.

```
int qIndex = i - 180;
if (qIndex >= 1 && qIndex < 360) {

// Save pressure rise rate array to analysis result
analysisResult->pressureRiseRate[qIndex] = pressureRiseRate;

// Calculate heat release rate in the combustion window based on cal (i.e -15° to + 30°)
if (crankAngle > heatReleaseAnalysisLowerLimitTheta &&
    crankAngle < heatReleaseAnalysisUpperLimitTheta) {
   double heatReleaseRate = newtonsPerBar * (gainA * volumeShifted[i]
       * pressureRiseRate + gainB * pressure *
       (volumeShifted[i]-volumeShifted[i-1])/deltaAngle);

  // Save hrr  array to analysis result
   analysisResult->heatReleaseRate[qIndex]  = heatReleaseRate;
   heatRelease[qIndex] = heatRelease[qIndex-1] + heatReleaseRate * 0.5;

   // Record max hrr
   if (heatRelease[qIndex] > maxHeatRelease) {
       maxHeatRelease = heatRelease[qIndex];
    }
} else {
       heatRelease[qIndex] = heatRelease[qIndex-1];
       analysisResult->heatReleaseRate[qIndex] = 0;
}
```

- Fourth, calculate gross IMEP.

```
// Compute Gross IMEP
double volumeCyl = (maxVolume - minVolume) * 1000.0;
double grossTQPerCyl = integralPdV / 2 / 2 / M_PI;
analysisResult->grossImep = (4 * M_PI * grossTQPerCyl / volumeCyl) / 100;
```

- Finally, CA50 is calculated. Note that if it falls between to two half degree angle, interpolation is applied.

```
// Find CA50
double midQ = 0.5 * maxHeatRelease;
analysisResult->heatRelease50 = midQ;
crankAngle = -89.5;
for (int i = 1; i < 360; ++i){
    if (heatRelease[i] >= midQ){
                double slope = (midQ - heatRelease[i-1])/(heatRelease[i] - heatRelease[i-1]);
        analysisResult->crankAngle50 = crankAngle - 0.5 + 0.5 * slope;

                return;
    }
    crankAngle += deltaAngle;
}
```

- analyseData360degATDC() – this function is called for every firing order at 360° after TDC and analyzes parameters for visualization only. Values can be used for non-critical feedback control. The default algorithm computes net MEP and pumping loss. The analysis results are saved into the same structure used by analyseData180degATDC(). Data is sent via ethernet port if the PC is connected. Below are highlights of the function.

- Similar to analyseData180degATDC(), the first major step is to filter the pressure array. The main difference that now the array contains 1440 points for the entire cycle (-360° to 360°).

```
// Filter pressure voltage array based on chosen filter
double* inputArray;
double adcPressureFiltered[720];
if (dataFilterType == THREE_POINT_MOVING_AVERGAE) { // 3-point filter
    threePointMovingAveragefilter(adcPressure, adcPressureFiltered, 720);
    inputArray = adcPressureFiltered;
} else if (dataFilterType == FIVE_POINT_MOVING_AVERGAE) { //5-point filter
    fivePointMovingAveragefilter(adcPressure, adcPressureFiltered, 720);
    inputArray = adcPressureFiltered;
} else {
    inputArray = adcPressure; // unfiltered
}
```

- Next, loop through the pressure array to find work, minimum and maximum volumes.

```
// Loop from -360° to +359.5° TDC @ 0.5° resolution
for (int i = 1; i < 1440; i++) {

    // Save pressure to analysis result
    double pressure = inputArray[i] * barsPerVolt + analysisResult->pressureShift;
    analysisResult->pressure[i] = pressure;

    // Calculate work
    integralPdV += pressure * (volume[i]-volume[i-1]) * newtonsPerBar;

    // Find min & max volume
    if (volume[i] < minVolume) {
        minVolume = volume[i];
    }
    if (volume[i] > maxVolume) {
        maxVolume = volume[i];
    }

    crankAngle += deltaAngle;
}
```

- Finally, calculate net IMEP and pumping loss.

```
// Compute Net IMEP and Pumping loss
double volumeCyl = (maxVolume - minVolume) * 1000.0;
double netTQPerCyl = integralPdV / 2 / 2 / M_PI;
analysisResult->netImep = (4 * M_PI * netTQPerCyl / volumeCyl) / 100;
analysisResult->pumpingLoss = analysisResult->grossImep - analysisResult->netImep;
```

- threePointMovingAveragefilter() – optional three-point pressure filters for pressure arrays. This function is called when analyzing the pressure array depending on the calibration settings.

- fivePointMovingAveragefilter() – optional five-point pressure filters for pressure arrays. Similar to threePointMovingAveragefilter(), this function is called when analyzing the pressure array depending on the calibration settings.

8.2 Modify the combustion analysis algorithm.

This section walks through the code involving the ringing intensity calculation into the analyszeData180degATDC() function. The two versions “Analysis.c” with and without ringing intensity calculation will be provided for reference. Please use keyword “ringing intensity” to find code sections with modifications.

Ringing intensity (RI) is calculated using the following equation:

$$RI = \frac{1}{2\gamma} \frac{\left(\beta \left(\frac{dP}{dt}\right)_{max}\right)^2}{P_{max}} \sqrt{\gamma\, R\, T_{max}}$$

where γ is the ratio of specific heats (c_p/c_v), β is a scaling coefficient, $\frac{dP}{dt}$ is the pressure rise rate over time (bar/ms), P_{max} is the max pressure (bar), R is the universal gas constant 287 (J/kg K), and T_{max} is max temperature (°C) derived from the ideal gas law.

First, before analysis begins, declare the constants and initialize variables. Insert the following at line 129.

```
// Variables for ringing intensity calculations
double riBeta = 0.05; // scaling coefficient
double riGamma = 1.39; // ratio of specific heats (cp/cv)
double riRconst = 287.0; // universal gas constant (J/kg K)
double intakeTemp = getAnalysisCal(INTAKE_TEMPERATURE); // get intake temperature from calibration
double riMass = 1.03 * 100000.0 * (volumeMax - volumeMin) \
/ riRconst / (intakeTemp + 273.0); // mass (kg) derived ideal gas law
double maxTemperature = -1000.0; // to record max temperature (K)
```

Next, when looping through the pressure array, calculate temperature and record T_{max}. At the meantime, compute $\frac{dP}{dt}$ and record its max. Note that $\frac{dP}{dt}$ is calculated from the pressure rise rate ($\frac{dp}{d\theta}$) and engine rpm, which are available at this point of the analysis. Insert the following at line 181.

```
// Calculate temperature & find max - ringing intensity
double temperature = pressure * volumeShifted[i] / riRconst / riMass;
if (temperature > maxTemperature) {maxTemperature = temperature;}
```

After the loop is done, the ringing intensity can be calculated, and it is saved to the analysis result. Insert the following at line 227.

```
// Calculate ringing intensity
double factor = 0.5 / riGamma / maxPressure * sqrt(riGamma * riRconst * \
(maxTemperature * 100000.0)) / 10.0; // combine all non-changing variables during the RI cal
for (int i = 0; i < 360; i++)
{
	double pressureRiseRateMs = analysisResult->pressureRiseRate[i] * \
		((double)DEGREES_IN_ENGINE_REV / (double)SECONDS_PER_MINUTE) * \
	analysisResult->engineSpeed/1000.0; // dpdt (bar/s)
	analysisResult->ringingIntensityArr[i] =  pow(riBeta * \
		pressureRiseRateMs, 2.0) * factor; // Save ringing intensity to analysis result
}
```

Finally, save the file as “Analysis.c” and use PC software to upload the algorithm to ECHO. Press the “Algorithm” button on the Setup page, as show in Figure 17.

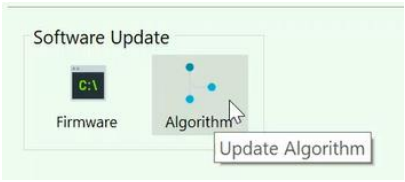


Figure 17: Update Algorithm button.

Select the “Analysis.c” file in the popup window as shown in Figure 18, and then click “Open”. The file will be uploaded to ECHO and recompiled with its embedded library. A result message will appear in the status window after the process.

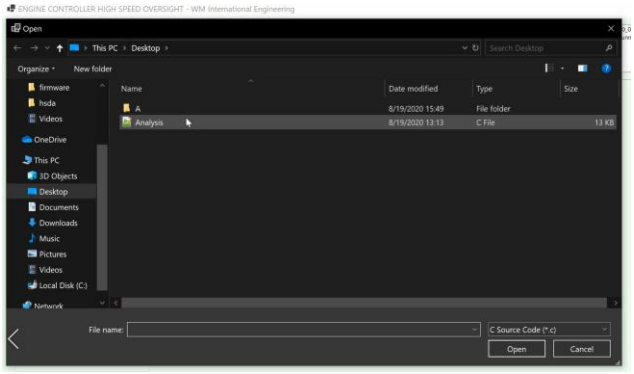


Figure 18: Pop-up window for selecting the "Analysis.c" file.

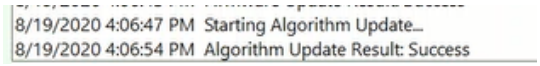


Figure 19: Status windows showing result message after an algorithm update.

To verify that added computation does not add significant latency to cycle-to-cycle, please send the modified algorithm to us for review. After that, the algorithm can be integrated to CAN 2.0 feedback and PC side software visual if needed.

9. Appendix A: HATCI and MTU Harness

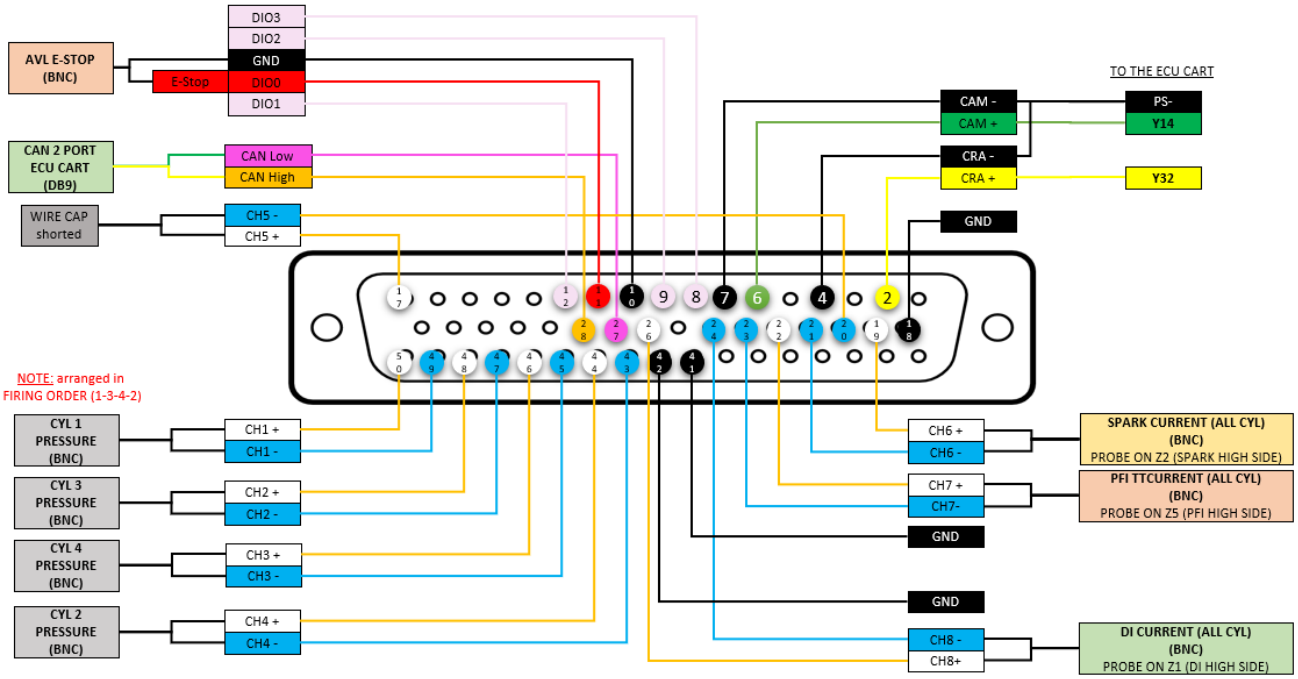


Figure A-1: Harness wiring for HATCI and MTU application.