1 Introduction

The XCP plug-in is a lightweight implementation of an XCP slave. It enables measuring and calibrating of the target ECU without impacting overall ECU’s performance or memory footprint.

The XCP plug-in interfaces between a measurement / calibration tool like CANoe or CANape and the rest of the tools stack down to the target ECU. All microcontrollers, supported by iSYSTEM tools, can be handled by the XCP plug-in.

The XCP plug-in enables the measurement and calibration of an ECU without the need of any slave code on the ECU, without performance degradation to the ECU operation and without the need of a physical interface (like CAN, FlexRAY, …), other than the ECUs CPU debug or trace ports.

The XCP plug-in supports polled mode access, where access is directed from the XCP master and also high speed data acquisition, where data accesses are directed by the emulator (iC5000, iC3000, …) and streamed to the XCP master.

1.1 Features

- Synchronous data acquisition (DAQ)
- Direct memory access (polling)
- Online memory calibration (read / write access)
- Timestamped data transfer, generation of event timestamps by the ECU/Emulator

1.2 Limitations

- Ethernet transport layer supported only
- Only dynamic DAQ list supported
- Bypassing is not supported
- Block mode is not supported
- Resume mode is not supported
- Checksum is not supported
- Dynamic event list (Plug-and-play) is not supported
### 1.3 Performance Characteristics

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp resolution</td>
<td>100 μs</td>
</tr>
<tr>
<td>Min sampling interval – polling</td>
<td>1 ms</td>
</tr>
<tr>
<td>Min sampling interval – DAQ</td>
<td>100 μs</td>
</tr>
<tr>
<td>Bypassing latency time</td>
<td>&lt; 500 μs</td>
</tr>
<tr>
<td>Max DAQ events</td>
<td>256</td>
</tr>
<tr>
<td>Max DAQ per Event</td>
<td>1</td>
</tr>
<tr>
<td>Max ODT entries</td>
<td>256</td>
</tr>
<tr>
<td>Max ODT entry size</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

1. ODT = Object Descriptor Table describes the mapping between the synchronous data transfer objects and the ECU’s memory. Handled by XCP master and contains one or more ODT entries.

2. An entry in an ODT references ECU’s data element by its address, the address extension and the size of the element. Handled by XCP master and usually maps to single observed variable.
2 Usage

The XCP master (i.e. CANoe, CANape) and the XCP slave (winIDEA XCP plugin) must be configured properly to establish communication. The communication protocol (TCP or UDP) and port should match. If the master and slave are in different host you will have to configure the addresses accordingly, by default the communication is setup for localhost connection.

2.1 General setup

1) Enable the XCP plug-in in winIDEA. (Plugins/Options)
2) Enable the Status Window of the XCP plugin. (Plugins/Options/XCP/Status window)

3) Review the XCP configuration options for both the master and the slave. Generally, the slave functions as a “listening” server that sources data and the master (i.e. CANoe) as a client that consumes it.

**XCP slave configuration:**

- **Protocol**: TCP
- **Port**: 5555
XCP master configuration (i.e. CANoe):
2.2 **Modes of operation**

The XCP plug-in enables ECU measurement (ECU’s memory observation) and ECU calibration (ECU’s memory write). In all cases, it is the master’s (CANoe, CANape) responsibility to set-up the measurement environment.

The ASAM 2MC configuration file (extension .A2L) is a good starting point to prepare the measurement environment. A default WINIDEA_XCPSERVER.A2L file is provided by iSYSTEM for easy start-up.

On the master side, data acquisition may be configured as DAQ or Polling. DAQ is recommended as data acquisition is much faster and the sampling interval more consistent.

2.2.1 **DAQ**

In DAQ mode, data acquisition is performed by the emulator. For maximum performance, disable »Real time memory-access« under **Debug/Options/ Memory Access**. If real-time memory access is enabled and in use, the debugger and the DAQ acquisition subsystem will compete resources, resulting in a slower or less stable sampling interval.
Example

DAQ configuration in CANoe:

In this case, iXcpArray_01 variable will be sampled by emulator at maximum sampling rate. Whenever variable changes, an event is triggered and data streamed to XCP master.

When DAQ is running, this is clearly indicated on the XCP’s plugin main page.
2.2.1.1 DAQ Performance issues

General performance is indicated in the line »DAQ average/desired sampling time (us)«. Desired sampling time comes from minimum required event sampling time (see table):

<table>
<thead>
<tr>
<th>DAQ event name</th>
<th>DAQ event number</th>
<th>Desired sampling rate (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_rate</td>
<td>0</td>
<td>max possible rate</td>
</tr>
<tr>
<td>1ms</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>10ms</td>
<td>2</td>
<td>10.000</td>
</tr>
<tr>
<td>100ms</td>
<td>3</td>
<td>100.000</td>
</tr>
<tr>
<td>1s</td>
<td>4</td>
<td>1.000.000</td>
</tr>
<tr>
<td>user defined</td>
<td>5-255</td>
<td>1.000.000</td>
</tr>
</tbody>
</table>

If average sampling time exceeds the desired minimum required sampling time, the status line appears in red, clearly warning the end user of irregular sampling conditions.

Please take care when deciding on event number. Events 0,1 and 5-255 cause a lot of traffic on the debug port which could lead to irregular measurement conditions.
2.2.2 Polling

When the polling mode is used the memory access requests are generated by the master. The XCP commands (read and/or write to ECU’s memory) are processed sequentially by the XCP slave. The statistics are clearly displayed in the “XCP statistics” section.

Note: Real-time memory access under Debug/Options/Memory Access must be ENABLED in this case.

Example

Polling configuration in CANoe:
In this case, every 100 ms a read request is forwarded to XCP plugin, which reads the ECU’s memory. Statistics about the operation of the plug-in in this mode are shown on the XCP plugin statistics section:

![XCP plugin statistics](image)

If real-time memory access is not enabled, access to ECU’s memory is disabled. Statistics under XCP memory reads total/errors shows the number of unsuccessful attempts. XCP_CMD_DENIED is returned to the master.
2.3 The XCP plugin window

- Command buttons to start, stop and configure the XCP plugin.
- The status line displays the general XCP plugin status (Stopped, Running / Listening).
- The configuration options display the general XCP plugin configuration (protocol, port, memory access mode, Autostart).
- The master (client) info displays the master connection status (whether a master is connected or not).
- The XCP statistics display overall XCP commands statistics. Memory reads, writes and errors are reported separately.
- The DAQ statistics display the overall DAQ acquisition performance.
2.4 Measurement sequence

1. Prepare measurement configuration on the master side. This includes adoption of memory locations and measurement mode for every observed variable. Start with WINIDEA_XCPSERVER.A2L template, which includes a single observed variable. Link ECU’s MAP file with A2L observation variables.

For DAQ-based measurement, following events/measurement modes are available by default:

- max_rate (max sampling rate on the Emulator)
- 1ms_loop (1ms sampling rate on the Emulator)
- 10ms_loop (10ms sampling rate on the Emulator)
- 100ms_loop (100ms sampling rate on the Emulator)
- 1s_loop (1s sampling rate on the Emulator)

Up to 256 events could be used (please add own to .A2L file).

Please note, that an event will ONLY be triggered, if ANY of the observed variable(s) has changed.

Picture: Example of measurement configuration in CANape

2. Prepare target application.
3. Run the XCP plugin.
4. Run the target application from winIDEA.
5. Start measurement from the master. Optionaly use debugger Run control to manipulate target execution.