# Hardware and Software Manual

| | | | | | |
|---|---|---|---|---|---|
| **Customer** | Internal | | | **Via** | |
| **Project** | STW_INC_3AXIS, j1939 3-Axis Inclinometer Library for CoDeSys3.5 | | | **ID** | -n/a- |
| **Revision** | 2 | **Software Version** | 2.0r0 | **Date** | 2020-03-11 |
| **Preparer** | STW/jcg | | | | |

STW Technic, LP
Mobile Controllers and Measurement Technologies

Pioneering new technologies
Pioneering new technologies

**Address**
Suite 260
3000 Northwoods Parkway
Norcross, GA 30071 – USA

**Telephone**
770-242-1002
**Fax**
770-242-1006

**E-mail**
info@stw-technic.com
**Internet**
www.stw-technic.com

# 1 Table of Contents

# 2 System Overview

The STW_INC_3AXIS library for CoDeSys v3.5 Programming allows easily handling of CANbus communication between an STW ECU (the ECU) and an STW j1939 3-axis Inclinometer (the Module). The library may be added to a CoDeSys v3.5 user project for the following platforms: ESX-3Xx and ESX-3Cx.

The library is organized into a set of folders, each containing related components. Most user interaction will be with the function blocks contained within the CAN_Handling and TX_Functions folders. Here the user will find functions and function blocks to handle all necessary transmission and reception of CAN messages.

The Constants folder contains a Global Constants list for various parameters required in the CAN messages. The user can access these constants when populating configuration structures or when reading received data.

The Initialization_Functions folder contains a function for initializing all CAN objects necessary to work with the transmission and reception function blocks.

The RX_Functions folder contains all support functions used internally by the receive function block. In general, the user will not need to access these.

The Structures folder contains definitions of all structures used by the various components of the library. Most likely the user will want to declare their own local instances of several of these structures.

The TX_Functions folder contains all transmit functions and all support functions used internally by the transmit functions.

## 2.1 Communication Overview

In the user's CoDeSys project, the user will need to first initialize the CANbus channels that will be used by the library. It is possible to use multiple CANbus channels if necessary, however each channel will require separate instances of all necessary function blocks. The CANbus channels can be initialized using the standard STW CANbus Configuration tabs in the Devices window. The Module communicates using the j1939 protocol.

The Module uses auto-baudrate detection at startup. The Module listens passively on the CAN bus and cycles through all possible bitrates until it detects traffic. Once traffic is detected, it configures itself for that bitrate and begins normal operation. The Module will not transmit any data until this detection has occurred, so it may be necessary for the user application to generate some form of "wakeup" message on the bus until the Module begins to communicate.

To initialize the necessary CAN objects for transmission and reception of CAN messages, the user can use the initialization function F_INC_Init_CAN_Objects. Alternatively, the user can initialize CAN objects manually using the CoDeSys Devices window.

The user can use the FB_INC_Receive function block to manage reception of all incoming CAN traffic from the Module. All data is returned to the user as a set of sensor structures. Within each structure is a set of individual data structures which are organized by j1939 PGN. The user should pay attention to the timeout information provided with each PGN structure. This will inform the user if the data has not been updated over a period defined by the library constants INC_TX_RATE_PGN_SSI_MS, INC_TX_RATE_PGN_BI_MS or INC_TX_RATE_RESPONSE_MS. The user can then decide if the data should be used or if new data should be requested. The timeout status of T_INC_PGN_SSI, or

T_INC_PGN_BI if configured for single axis operation, can be used to determine overall health of the Module since this message is automatically transmitted at a fixed rate.

The user can use the various Transmit functions to transmit the associated CAN messages to the Module. Each function will transmit one message immediately and only once per execution, so it is the user's responsibility to properly schedule their execution.

The general flow of operation will be the following:

1. User initializes CANbus channel.

2. User calls F_INC_Init_CAN_Objects once to configure the necessary CAN RX and TX objects.

3. User begins calling an instance of FB_INC_Recive cyclically to receive all incoming data.

4. FB_INC_Receive reports reception of T_INC_PGN_SSI and T_INC_PGN_BI for each sensor channel from the Module.

5. If both of the above messages are timed out, then Module is not transmitting on the CANbus. The user must generate a 'wakeup' message (any length, any content) at the desired bitrate until the Module begins transmitting.

6. User calls F_INC_TX_REQUEST to read the various configuration and data for each sensor.

7. If configuration is necessary, the user calls F_INC_TX_SERVICE once for each sensor channel to set the Module into service mode. The service enable code is saved in the library as the constant INC_SERVICE_MODE_ENABLE.

8. User calls F_INC_TX_ADDR, F_INC_TX_DEV_SET, and F_INC_TX_OFFSET_ADJ if needed to configure the Module. If a valid configuration had previously been saved to non-volatile memory on the Module, this is not necessary.

9. Once configuration is complete, the user calls F_INC_TX_SERVICE once for each sensor channel to disable service mode. The service disable code is saved in the library as the constant INC_SERVICE_MODE_DISABLE.

# 3 Software

| | | | | | |
|---|---|---|---|---|---|
| **File name** | STW_INC_3AXIS.library | **Load Order** | 1 | **of** | 1 |
| **Description** | CoDeSys v3.5 library for STW j1939 3-Axis Inclinometer communication | | | | |
| **Load Tool** | CoDeSys v3.5 SP6 | | | | |
| **Options** | Hardware Specific | ESX-3Xx, ESX-3Cx | | | |
| | Firmware Specific | - | | | |
| **Notes** | | | | | |

## 3.1 Structures

All user data for the Module will be written to/from the CANbus via a series of structures, which can be found within the Structures directory. The user should declare an instance (most likely globally scoped) of each structure as needed to store this data.

```
TYPE T_MESSAGE_STATUS :
STRUCT
   q_Timeout : BOOL; // TRUE = Timeout has occurred ( > 4x TX Rate)
   u32_RX_Time_ms : UDINT; // Receive timestamp in ms
END_STRUCT
END_TYPE
```

All j1939 PGN message structures detailed below include an instance of **T_MESSAGE_STATUS**. During message reception handling, this structure is populated with a timestamp value and a timeout flag. These values can be used to determine if received data is relevant and/or if the Module is active on the CANbus. During message transmission, this structure is not used.

### 3.1.1 Sensor Structures

```
TYPE T_INC_SENSOR :
STRUCT
   t_SSI : T_INC_PGN_SSI;
   t_BI : T_INC_PGN_BI;
   t_ACK : T_INC_PGN_ACK;
   t_REV_SNR : T_INC_PGN_REV_SNR;
   t_SETTINGS : T_INC_PGN_SETTINGS;
   t_OFFSETS : T_INC_PGN_OFFSETS;
   t_ACCEL : T_INC_PGN_ACCEL;
   t_GYRO : T_INC_PGN_GYRO;
   t_TEMP : T_INC_PGN_TEMP;
   t_ACCEL_ANGLE : T_INC_PGN_ACCEL_ANGLE;
   t_CAL_ACCEL : T_INC_PGN_CAL_ACCEL;
END_STRUCT
END_TYPE
```

Contains all data received from one of the Module's primary sensor channels (Sensor 1 or Sensor 2). Each individual PGN structure is populated with data as the respective message is received.

```
TYPE T_INC_COMPENSATED_SENSOR :
STRUCT
   t_SSI : T_INC_PGN_SSI;
   t_BI : T_INC_PGN_BI;
   t_ACK : T_INC_PGN_ACK;
   t_REV_SNR : T_INC_PGN_REV_SNR;
   t_SETTINGS : T_INC_PGN_SETTINGS;
END_STRUCT
END_TYPE
```

Contains all data received from one of the Module's compensated sensor channel. Each individual PGN structure is populated with data as the respective message is received.

### 3.1.2 Receive Structures

```
TYPE T_INC_PGN_SSI :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   u16_Pitch_Angle : UINT;
   u16_Roll_Angle : UINT;
   u16_Pitch_Rate : UINT;
   u8_Pitch_Angle_Figure_of_Merit : USINT;
   u8_Roll_Angle_Figure_of_Merit : USINT;
   u8_Pitch_Rate_Figure_of_Merit : USINT;
   u8_Pitch_and_Roll_Compensated : USINT;
   u8_Pitch_and_Roll_Measurement_Latency : USINT;

   // scaled values
   f32_Pitch_Angle_deg : REAL;
   f32_Roll_Angle_deg : REAL;
   f32_Pitch_Rate_deg_sec : REAL;
   f32_Pitch_and_Roll_Measurement_Latency_ms : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 PGN 61459 “Slope Sensor Information”. This PGN is only transmitted by the Module when configured for Dual-Axis mode.
Unscaled values:

- **u16_Pitch_Angle** – Rotation about the x-axis. SPN 3318, 0.002 deg/bit, -64 offset
- **u16_Roll_Angle** – Rotation about the y-axis. SPN 3319, 0.002 deg/bit, -64 offset
- **u16_Pitch_Rate** – Pitch rate of change. SPN 3322, 0.002 degrees/sec/bit, -64 degree/sec offset
- **u8_Pitch_Angle_Figure_of_Merit** – Pitch figure of merit. SPN 3323, 4 states/2 bit, 0 offset
- **u8_Roll_Angle_Figure_of_Merit** – Roll figure of merit. SPN 3324, 4 states/2 bit, 0 offset
- **u8_Pitch_Rate_Figure_of_Merit** – Pitch rate figure of merit. SPN 3325, 4 states/2 bit, 0 offset
- **u8_Pitch_and_Roll_Compensated** – Compensation mode. SPN 3326, 4 states/2 bit, 0 offset
- **u8_Pitch_and_Roll_Measurement_Latency** – Measurement latency. SPN 3327, 0.5 ms/bit, 0 offset

Scaled values:

- **f32_Pitch_Angle_deg** – Pitch angle in degrees.
- **f32_Roll_Angle_deg** – Roll angle in degrees.
- **f32_Pitch_Rate_deg_sec** – Pitch rate in degrees per second.
- **f32_Pitch_and_Roll_Measurement_Latency_ms** – Measurement latency in milliseconds.

```
TYPE T_INC_PGN_BI :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   u16_Relative_Blade_Height : UINT;
   u16_Blade_Rotation_Angle : UINT;
   u16_Measurement_Latency : UINT;
   u8_Relative_Blade_Height_Figure_of_Merit : USINT;
   u8_Blade_Rotation_Angle_Figure_of_Merit : USINT;

   // scaled values
   f32_Relative_Blade_Height_mm : REAL;
   f32_Blade_Rotation_Angle_deg : REAL;
   f32_Measurement_Latency_ms : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 PGN 61460 “Blade Information (Ladder Angle)”. This PGN is only transmitted by the Module when configured for Single-Axis mode.
Unscaled values:

- **u16_Relative_Blade_Height** – Distance from ground reference. SPN 3365, 0.1 mm/bit, -3200 mm offset
- **u16_Blade_Rotation_Angle**– Rotation about the z-axis. SPN 3331, 1/128 deg/bit, -200 deg offset
- **u16_Measurement_Latency** – Measurement latency. SPN 3336, 0.5 ms/bit, 0 offset
- **u8_Relative_Blade_Height_Figure_of_Merit** – Height figure of merit SPN 3367, 4 states/2 bit, 0 offset
- **u8_Blade_Rotation_Angle_Figure_of_Merit** – Rotation figure of merit. SPN 3332, 4 states/2 bit, 0 offset

Scaled values:

- **f32_Relative_Blade_Height_mm** – Relative blade height in millimeters.
- **f32_Blade_Rotation_Angle_deg** – Blade rotation angle in degrees.
- **f32_Measurement_Latency_ms** – Measurement latency in milliseconds.

```
TYPE T_INC_PGN_ACK :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   au8_Ack_Data : ARRAY [0..7] OF BYTE;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 65468 “Acknowledge Command”. This PGN is transmitted by the Module as a response to configuration commands. The byte array contains the data echo of the configuration command message.

```
TYPE T_INC_PGN_REV_SNR :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   u8_Revision : USINT;
   u32_Serial_Number : UDINT;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61185 “Revision and Serial Number”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x01, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **u8_Revision** – The source code revision of the device. This value is transmitted in ASCII: 0x41 to 0x5A (ASCII A to ASCII Z).
- **u32_Serial_Number** – The serial number of the device: 0 to 0xFFFFFFFE.

```
TYPE T_INC_PGN_SETTINGS :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   u8_Pitch_Polarity : USINT;
   u8_Roll_Polarity : USINT;
   u8_Pitch_and_Roll_Orientation : USINT;
   u8_Vibe_Filter : USINT;
   u8_Vertical_Plane : USINT;
   u8_Sensor_Data_Transmitted : USINT;
   u8_Sample_Size : USINT;
   u8_Accelerometer_Increment : USINT;

   // scaled values
   f32_Accelerometer_Increment_pct : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61186 “Stored Settings Data”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x02, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **u8_Pitch_Polarity** – The stored polarity of the pitch rotation. The default is CCW = positive. 3 states/2 bit, 0 offset
- **u8_Roll_Polarity** – The stored polarity of the roll rotation. The default is CCW = positive. 3 states/2 bit, 0 offset
- **u8_Pitch_and_Roll_Orientation** – The stored pitch and roll axis orientation. The default is standard orientation. 3 states/2 bit, 0 offset
- **u8_Vibe_Filter** – The stored vibration filter setting. The default is noise block mode. 3 states/2 bit, 0 offset
- **u8_Vertical_Plane** – The stored vertical plane of the device. The default is the Z-Plane. 7 states/3 bit, 0 offset
- **u8_Sensor_Data_Transmitted** – The stored selection for which sensors are currently transmitting the ‘Slope Sensor Information’ message. 7 states/3 bit, 0 offset
- **u8_Sample_Size** – The stored number of measurements that are averaged together when calculating the angle. The default is 20. 1 sample/bit, 0 offset, range 0-20
- **u8_Accelerometer_Increment** – The stored value for the percentage at which the accelerometer angle is added to the reported angle. The default is 1%. 0.1 %/bit, 0 offset, range 0.1%-10%

Scaled values:

- **f32_Accelerometer_Increment_pct** – Accelerometer Increment in percent.

```
TYPE T_INC_PGN_OFFSETS :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   s16_Pitch_Offset : INT;
   s16_Roll_Offset : INT;
   u16_Zero_Pitch_Offset : UINT;
   u16_Zero_Roll_Offset : UINT;

   // scaled values
   f32_Pitch_Offset_deg : REAL;
   f32_Roll_Offset_deg : REAL;
   f32_Zero_Pitch_Offset_deg : REAL;
   f32_Zero_Roll_Offset_deg : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61187 “Stored Offsets Data”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x03, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **s16_Pitch_Offset** – The stored adjustment to the pitch offset. This value is expressed as a 16-bit word in two’s compliment. 0.002 deg/bit, 0 offset, range -65.634 to 65.534
- **s16_Roll_Offset** – The stored adjustment to the roll offset. This value is expressed as a 16-bit word in two’s compliment. 0.002 deg/bit, 0 offset, range -65.634 to 65.534
- **u16_Zero_Pitch_Offset** – The stored user set offset used to zero the pitch angle. 0.002 degrees/sec/bit, -64 degree/sec offset
- **u16_Zero_Roll_Offset** – The stored user set offset used to zero the roll angle. 0.002 degrees/sec/bit, -64 degree/sec offset

Scaled values:

- **f32_Pitch_Offset_deg** – Pitch offset in degrees.
- **f32_Roll_Offset_deg** – Roll offset in degrees.
- **f32_Zero_Pitch_Offset_deg** – Zero pitch offset in degrees.
- **f32_Zero_Roll_Offset_deg** – Zero roll offset in degrees.

```
TYPE T_INC_PGN_ACCEL :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   s16_Raw_X_Acceleration : INT;
   s16_Raw_Y_Acceleration : INT;
   s16_Raw_Z_Acceleration : INT;

   // scaled values
   f32_Raw_X_Acceleration_mg : REAL;
   f32_Raw_Y_Acceleration_mg : REAL;
   f32_Raw_Z_Acceleration_mg : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61188 “Raw Acceleration Data”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x04, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **s16_Raw_X_Acceleration** – The linear acceleration sensor’s raw x-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g
- **s16_Raw_Y_Acceleration** – The linear acceleration sensor’s raw y-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g
- **s16_Raw_Z_Acceleration** – The linear acceleration sensor’s raw z-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g

Scaled values:

- **f32_Raw_X_Acceleration_mg** – Raw x-axis acceleration in milli-g.
- **f32_Raw_Y_Acceleration_mg** – Raw y-axis acceleration in milli-g.
- **f32_Raw_Z_Acceleration_mg** – Raw z-axis acceleration in milli-g.

```
TYPE T_INC_PGN_GYRO :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   s16_Raw_Gyro_Pitch : INT;
   s16_Raw_Gyro_Roll : INT;
   s16_Raw_Gyro_Yaw : INT;

   // scaled values
   f32_Raw_Gyro_Pitch_mdps : REAL;
   f32_Raw_Gyro_Roll_mdps : REAL;
   f32_Raw_Gyro_Yaw_mdps : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61189 “Raw Gyroscope Data”. This PGN is transmitted by the Module as a response to a data request command.
CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x05, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.
Unscaled values:

- **s16_Raw_Gyro_Pitch** – The angular rate sensor’s pitch axis (X) value. This value is expressed as a 16-bit word in two’s compliment. 35mdps/bit, 0 offset, range -1000dps to 1000dps
- **s16_Raw_Gyro_Roll** – The angular rate sensor’s roll axis (Y) value. This value is expressed as a 16-bit word in two’s compliment. 35mdps/bit, 0 offset, range -1000dps to 1000dps
- **s16_Raw_Gyro_Yaw** – The angular rate sensor’s yaw axis (Z) value. This value is expressed as a 16-bit word in two’s compliment. 35mdps/bit, 0 offset, range -1000dps to 1000dps

Scaled values:

- **f32_Raw_Gyro_Pitch_mdps** – Raw gyroscope pitch in millidegrees per second.
- **f32_Raw_Gyro_Roll_mdps** – Raw gyroscope roll in millidegrees per second.
- **f32_Raw_Gyro_Yaw_mdps** – Raw gyroscope yaw in millidegrees per second.

```
TYPE T_INC_PGN_TEMP :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   s16_Raw_Temperature : INT;

   // scaled values
   f32_Raw_Temperature_degC : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61190 “Temperature Data”. This PGN is transmitted by the Module as a response to a data request command.
CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x06, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.
Unscaled values:

- **s16_Raw_Temperature** – The sensor’s temperature data. This value is expressed as a 16-bit word in two’s compliment. 0.0039 degC/bit, 25 degC offset, range -102.996 degC to 152.996 degC

Scaled values:

- **f32_Raw_Temperature_degC** – Raw temperature in degrees Celsius.

```
TYPE T_INC_PGN_ACCEL_ANGLE :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   u16_Accelerometer_Pitch_Angle : UINT;
   u16_Accelerometer_Roll_Angle : UINT;

   // scaled values
   f32_Accelerometer_Pitch_Angle_deg : REAL;
   f32_Accelerometer_Roll_Angle_deg : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61191 “Accelerometer Angle Data”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x07, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **u16_Accelerometer_Pitch_Angle** – The accelerometer angle between the vehicle’s y-axis and the ground plane, 0.002 deg/bit, -64 offset
- **u16_Accelerometer_Roll_Angle** – The accelerometer angle between the vehicle’s x-axis and the ground plane, 0.002 deg/bit, -64 offset

Scaled values:

- **f32_Accelerometer_Pitch_Angle_deg** – Accelerometer pitch angle in degrees.
- **f32_Accelerometer_Roll_Angle_deg** – Accelerometer roll angle in degrees.

```
TYPE T_INC_PGN_CAL_ACCEL :
STRUCT
   t_Msg_Status : T_MESSAGE_STATUS;

   // unscaled values
   s16_Cal_X_Acceleration : INT;
   s16_Cal_Y_Acceleration : INT;
   s16_Cal_Z_Acceleration : INT;

   // scaled values
   f32_Cal_X_Acceleration_mg : REAL;
   f32_Cal_Y_Acceleration_mg : REAL;
   f32_Cal_Z_Acceleration_mg : REAL;
END_STRUCT
END_TYPE
```

Contains all data received from j1939 proprietary PGN 61192 “Calibrated Acceleration Data”. This PGN is transmitted by the Module as a response to a data request command.

CAUTION – This PGN uses the Proprietary A peer-to-peer format with a Destination Address of 0x08, however the Module has not necessarily claimed this address. This may cause unexpected behavior in a j1939 network.

Unscaled values:

- **s16_Cal_X_Acceleration** – The linear acceleration sensor’s calibrated x-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g
- **s16_Cal_Y_Acceleration** – The linear acceleration sensor’s calibrated y-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g
- **s16_Cal_Z_Acceleration** – The linear acceleration sensor’s calibrated z-axis value. This value is expressed as a 16-bit word in two’s compliment. 0.061 mg/bit, 0 offset, range -1.99885g to 1.99885g

Scaled values:

- **f32_Cal_X_Acceleration_mg** – Calibrated x-axis acceleration in milli-g.
- **f32_Cal_Y_Acceleration_mg** – Calibrated y-axis acceleration in milli-g.
- **f32_Cal_Z_Acceleration_mg** – Calibrated z-axis acceleration in milli-g.

## 3.2 Function Blocks

### 3.2.1 Receive Function Blocks

```
FUNCTION_BLOCK FB_INC_Receive
VAR_INPUT
   ou32_RX_Handle_Sensor_01 : UDINT;
   ou32_RX_Handle_Sensor_02 : UDINT;
   ou32_RX_Handle_Compensated_Sensor : UDINT;
   ou32_PGN_SSI_TX_Rate : UDINT  := INC_TX_RATE_PGN_SSI_MS;
   ou32_PGN_BI_TX_Rate : UDINT   := INC_TX_RATE_PGN_BI_MS;
   ou32_Response_TX_Rate : UDINT := INC_TX_RATE_RESPONSE_MS;
END_VAR
VAR_OUTPUT
   ot_Sensor_01 : T_INC_SENSOR;
   ot_Sensor_02 : T_INC_SENSOR;
   ot_Compensated_Sensor : T_INC_COMPENSATED_SENSOR;
   os16_Status : INT := C_NO_ERR;
END_VAR
```

**- Inputs –**

- **ou32_RX_Handle_Sensor_01** – Handle of the RX object for Sensor 01 messages, created by F_HC100_Init_CAN_Objects or manually by the user application.
- **ou32_RX_Handle_Sensor_02** – Handle of the RX object for Sensor 02 messages, created by F_HC100_Init_CAN_Objects or manually by the user application.
- **ou32_RX_Handle_Compensated_Sensor** – Handle of the RX object for Compensated Sensor Data messages, created by F_HC100_Init_CAN_Objects or manually by the user application.
- **ou32_PGN_SSI_TX_Rate** – Message transmission rate. Default value INC_TX_RATE_PGN_SSI_MS.
- **ou32_PGN_BI_TX_Rate** – Message transmission rate. Default value INC_TX_RATE_PGN_BI_MS.
- **ou32_Response_TX_Rate** – Message transmission rate. Default value INC_TX_RATE_RESPONSE_MS.

**- Outputs -**

- **ot_Sensor_01** – Collation of all data received from sensor channel 1 to structure **T_INC_SENSOR**
- **ot_Sensor_02** – Collation of all data received from sensor channel 2 to structure **T_INC_SENSOR**
- **ot_Compensated_Sensor** – Collation of all data received from compensated sensor channel to structure **T_INC_COMPENSATED_SENSOR**
- **os16_Status** – Function block execution status. Any value other than C_NO_ERR indicates that one of the internal function or function block calls experienced an error.

**- Usage -**

This function block should be executed by the user application cyclically. It serves as the message reception engine and the user's access point for all incoming data. This function block accesses the CAN receive buffers (previously initialized by F_INC_Init_CAN_Objects or manually by the user application) and extracts and decodes all incoming CAN messages. It also determines the timeout state of all receive messages. All data is then stored to the appropriate structures and returned to the user.

## 3.3 Functions

### 3.3.1 Initialization Functions

```
FUNCTION F_INC_Init_CAN_Objects : INT
VAR_INPUT
   ou16_Channel : UINT;
   ou8_Sensor_01_SA : USINT;
   ou8_Sensor_02_SA : USINT;
   ou8_Compensated_Sensor_SA : USINT;
   opu32_RX_Handle_Sensor_01 : POINTER TO UDINT;
   opu32_RX_Handle_Sensor_02 : POINTER TO UDINT;
   opu32_RX_Handle_Compensated_Sensor : POINTER TO UDINT;
   opu32_TX_Handle : POINTER TO UDINT;
END_VAR
```

**- Inputs -**

- **ou16_Channel** – The ECU's CANbus channel that will be used to communicate with the Module.
- **ou8_Sensor_01_SA** – The Module's sensor 01 source address, default 0xE2
- **ou8_Sensor_02_SA** – The Module's sensor 02 source address, default 0xE3
- **ou8_Compensated_Sensor_SA** – The Module's Compensated Sensor Data source address, default 0xEA
- **opu32_RX_Handle_Sensor_01** – Pointer to the user's location for storing the handle of the created RX object for Sensor 01 messages
- **opu32_RX_Handle_Sensor_02** – Pointer to the user's location for storing the handle of the created RX object for Sensor 02 messages
- **opu32_RX_Handle_Compensated_Sensor** – Pointer to the user's location for storing the handle of the created RX object for Compensated Sensor Data messages
- **opu32_TX_Handle** – Pointer to the user's location for storing the handle of the created TX object for transmitting messages

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_RANGE** – Invalid CANbus channel
- **C_NOACT** – Invalid pointer
- **C_DEFAULT** – Error executing FB_x_can_obj_init or FB_x_can_obj_set_filter

**- Usage -**

Execution of this function is not required, but it is highly recommended. If it is not used, the user must manually configure all necessary CAN objects for transmitting and receiving the expected data.
This function initializes four CAN objects for receiving and transmitting data between the ECU and the Module. The handle for each CAN object is returned to the user's set of pointers. These handles are used by the transmit functions and receive function blocks to access the CAN network.

### 3.3.2 Receive Functions

Receive functions are all used internally by the FB_INC_Receive function block, so it is usually not necessary for the user to manually access these functions. However, they are available if the user wishes to use them to decode CAN messages. In all cases, if a decoding function returns a status value of **C_NOACT**, the function is not the appropriate one to decode the received CAN message. This is determined by the PGN within the ID of the message.

```
FUNCTION F_Decode_ID : T_J1939_ID
VAR_INPUT
   ot_CAN_ID : T_x_can_id;
END_VAR
```

**- Inputs -**

- **ot_CAN_ID** – The received CAN ID structure T_x_can_id to be decoded.

**- Return -**

- **T_J1939_ID** – J1939 CAN message ID structure

**- Usage -**

This is a helper function used internally by decode functions to extract j1939 protocol information from a received CAN ID structure. Note that this function will not return data if the ID is not in extended 29-bit format.

```
FUNCTION F_INC_Check_Compensated_Sensor_RX_Timeouts : INT
VAR_INPUT
    opt_Sensor : POINTER TO T_INC_COMPENSATED_SENSOR;
    ou32_PGN_SSI_TX_Rate : UDINT;
    ou32_PGN_BI_TX_Rate : UDINT;
    ou32_Response_TX_Rate : UDINT;
END_VAR
```

**- Inputs -**

- **opt_Sensor** – Pointer to user's copy of T_INC_COMPENSATED_SENSOR data.
- **ou32_PGN_SSI_TX_Rate** – Expected transmit rate of SSI messages.
- **ou32_PGN_BI_TX_Rate** – Expected transmit rate of BI messages.
- **ou32_PGN_Response_TX_Rate** – Expected response rate of requested messages.

**- Return -**

- **C_NO_ERR** – Function executed without error

**- Usage -**

Execution of this function by the user is not required, it is executed internally by FB_INC_Receive. This function compares the last receive timestamp of each expected message from the Compensated Sensor channel against the specified transmit rates. If the time difference exceeds 4x the timeout value, the data is tagged as timed out.

```
FUNCTION F_INC_Check_Sensor_RX_Timeouts : INT
VAR_INPUT
    opt_Sensor : POINTER TO T_INC_SENSOR;
    ou32_PGN_SSI_TX_Rate : UDINT;
    ou32_PGN_BI_TX_Rate : UDINT;
    ou32_Response_TX_Rate : UDINT;
END_VAR
```

**- Inputs -**

- **opt_Sensor** – Pointer to user's copy of T_INC_SENSOR data.
- **ou32_PGN_SSI_TX_Rate** – Expected transmit rate of SSI messages.
- **ou32_PGN_BI_TX_Rate** – Expected transmit rate of BI messages.
- **ou32_PGN_Response_TX_Rate** – Expected response rate of requested messages.

**- Return -**

- **C_NO_ERR** – Function executed without error

**- Usage -**

Execution of this function by the user is not required, it is executed internally by FB_INC_Receive. This function compares the last receive timestamp of each expected message from the Sensor 01 or 02 channel against the specified transmit rates. If the time difference exceeds 4x the timeout value, the data is tagged as timed out.

```
FUNCTION F_INC_Decode_ACCEL : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_ACCEL : POINTER TO T_INC_PGN_ACCEL;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_ACCEL** – Pointer to user's copy of T_INC_PGN_ACCEL data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Raw Acceleration Data" messages.

```
FUNCTION F_INC_Decode_ACCEL_ANGLE : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_ACCEL_ANGLE : POINTER TO T_INC_PGN_ACCEL_ANGLE;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_ACCEL_ANGLE** – Pointer to user's copy of T_INC_PGN_ACCEL_ANGLE data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Accelerometer Angle Data" messages.

```
FUNCTION F_INC_Decode_ACK : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_ACK : POINTER TO T_INC_PGN_ACK;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_ACK** – Pointer to user's copy of T_INC_PGN_ACK data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Acknowledge Command" messages.

```
FUNCTION F_INC_Decode_BI : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_BI : POINTER TO T_INC_PGN_BI;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_BI** – Pointer to user's copy of T_INC_PGN_BI data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Blade Information (Ladder Angle)" messages.

```
FUNCTION F_INC_Decode_CAL_ACCEL : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_CAL_ACCEL : POINTER TO T_INC_PGN_CAL_ACCEL;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_CAL_ACCEL** – Pointer to user's copy of T_INC_PGN_CAL_ACCEL data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Calibrated Acceleration Data" messages.

```
FUNCTION F_INC_Decode_GYRO : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_CAL_GYRO : POINTER TO T_INC_PGN_GYRO;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_GYRO** – Pointer to user's copy of T_INC_PGN_GYRO data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Raw Gyroscope Data" messages.

```
FUNCTION F_INC_Decode_OFFSETS : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_CAL_OFFSETS : POINTER TO T_INC_PGN_OFFSETS;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_OFFSETS** – Pointer to user's copy of T_INC_PGN_OFFSETS data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Stored Offsets Data" messages.

```
FUNCTION F_INC_Decode_REV_SNR : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_CAL_REV_SNR : POINTER TO T_INC_PGN_REV_SNR;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_REV_SNR** – Pointer to user's copy of T_INC_PGN_REV_SNR data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Revision and Serial Number" messages.

```
FUNCTION F_INC_Decode_SETTINGS : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_CAL_SETTINGS : POINTER TO T_INC_PGN_SETTINGS;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_SETTINGS** – Pointer to user's copy of T_INC_PGN_SETTINGS data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Stored Settings Data" messages.

```
FUNCTION F_INC_Decode_SSI : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_SSI : POINTER TO T_INC_PGN_SSI;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_SSI** – Pointer to user's copy of T_INC_PGN_SSI data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Slope Sensor Information" messages.

```
FUNCTION F_INC_Decode_TEMP : INT
VAR_INPUT
   ot_can_msg : T_x_can_msg;
   opt_PGN_TEMP : POINTER TO T_INC_PGN_TEMP;
END_VAR
```

**- Inputs -**

- **ot_can_msg** – The raw CAN message to be decoded.
- **opt_PGN_TEMP** – Pointer to user's copy of T_INC_PGN_TEMP data.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – This is not the correct function to decode this message (PGN mismatch)

**- Usage -**

Execution of this function by the user is required only if they are manually receiving CAN traffic instead of using FB_INC_Receive. This function is executed internally by FB_INC_Receive to decode incoming "Temperature Data" messages.

### 3.3.3 Transmit Functions

Each transmit function is paired with an encoding function that it uses internally to construct CAN messages from the constituent parts. It is usually not necessary for the user to manually access these encoding functions. However, they are available if the user wishes to use them to encode CAN messages. In all cases, if an encoding function returns a status value of **C_RANGE**, some value input into the function is outside the valid range for that parameter and the function does not continue encoding the message. In all cases, if a transmit function returns a status value of **C_NOACT**, the function's internal encoding function was unable to successfully build the message and therefore the message has not been transmitted. Otherwise, the function will return the status of the internal function block call FB_x_can_obj_send_msg.

```
FUNCTION F_Encode_ID : T_x_can_id
VAR_INPUT
   ou8_SA : USINT;
   ou16_PGN : UINT;
   ou8_DA : USINT;
   ou8_Priority : USINT;
END_VAR
```

**- Inputs -**

- **ou8_SA** – The j1939 source address.
- **ou16_PGN** – The j1939 PGN.
- **ou8_DA** – The j1939 destination address.
- **ou8_Priority** – The j1939 priority.

**- Return -**

- **T_x_can_id** – STW CAN message ID structure

**- Usage -**

This is a helper function used internally by transmit functions to build a 29-bit CAN ID structure from the basic component parts.

```
FUNCTION F_Encode_J1939_ID : T_x_can_id
VAR_INPUT
   ou8_Priority : USINT;
   ou8_Extended_Data_Page : USINT;
   ou8_Data_Page : USINT;
   ou8_PDU_Format : USINT;
   ou8_PDU_Specific : USINT;
   ou8_Source_Address : USINT;
END_VAR
```

**- Inputs -**

- **ou8_Priority** – The j1939 Priority.
- **ou8_Extended_Data_Page** – The j1939 Extended Data Page.
- **ou8_Data_Page** – The j1939 Data Page.
- **ou8_PDU_Format** – The j1939 PDU Format components of the PGN.
- **ou8_PDU_Specific** – The j1939 PDU Specific components of the PGN.
- **ou8_PDU_Source_Address** – The j1939 Source Address.

**- Return -**

- **T_x_can_id** – STW CAN message ID structure

**- Usage -**

This is a more precise helper function used internally by F_Encode_ID to build a 29-bit CAN ID structure from all component parts.

```
FUNCTION F_INC_Encode_ADDR : INT
VAR_INPUT
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou8_New_Sensor_SA : USINT;
   opt_CAN_Msg : POINTER TO T_x_can_msg;
END_VAR
```

**- Inputs -**

- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The current source address of the Module's sensor channel. Range 0-254
- **ou8_New_Sensor_SA** – The new source address of the Module's sensor channel. Range 0-254
- **opt_CAN_Msg** – Pointer to user's copy of T_x_can_msg.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_RANGE** – Invalid Sensor source address or New Sensor source address

**- Usage -**

Execution of this function by the user is required only if they are manually transmitting CAN traffic instead of using F_INC_TX_ADDR. This function is executed internally by F_INC_TX_ADDR to encode outgoing "Sensor Address Command" messages.

```
FUNCTION F_INC_TX_ADDR : INT
VAR_INPUT
   ou32_TX_Handle : UDINT;
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou8_New_Sensor_SA : USINT;
END_VAR
```

**- Inputs -**

- **ou32_TX_Handle** – Handle of the TX object for transmitting messages, created by F_INC_Init_CAN_Objects or manually by the user application.
- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The current source address of the Module's sensor channel. Range 0-254
- **ou8_New_Sensor_SA** – The new source address of the Module's sensor channel. Range 0-254

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – Error in the internal encoding function, message not sent
- **C_…** – Status return of the internal transmit function block FB_x_can_obj_send_msg, message may have not been sent

**- Usage -**

This function must be called by the user to transmit the "Sensor Address Command" message to one of the Module's sensor channels. Note: The Module must be in Service Mode to react to configuration command messages.

```
FUNCTION F_INC_Encode_DEV_SET : INT
VAR_INPUT
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou8_Sample_Size : USINT
   oq_Zero_Sensors : BOOL;
   oq_Detect_Vertical_Plane : BOOL;
   oq_Clear_Offset_Adjustment : BOOL;
   oq_Clear_Zero_Offset : BOOL;
   ou8_Axis_Mode : USINT;
   ou8_Pitch_Polarity : USINT;
   ou8_Roll_Polarity : USINT;
   ou8_Pitch_and_Roll_Orientation : USINT;
   ou8_Vertical_Plane : USINT;
   ou8_Vibe_Filter : USINT;
   ou8_Sensor_Data_Transmitted : USINT;
   ou16_Transmit_Interval_ms : UINT;
   of32_Accelerometer_Increment_pct : REAL;
   opt_CAN_Msg : POINTER TO T_x_can_msg;
END_VAR
```

**- Inputs -**

- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **ou8_Sample_Size** – The number of measurements that are averaged together when calculating the angle. The default is 20. Range 1-20
- **oq_Zero_Sensors** – TRUE = Set the current position of the device as zero for the pitch and roll
- **oq_Detect_Vertical_Plane** – TRUE = Automatically detect and set the vertical plane of the device
- **oq_Clear_Offset_Adjustment** – TRUE = Set the user defined pitch and roll offset adjustments to zero
- **oq_Clear_Zero_Offset** – TRUE = Clear the offset that was used to zero the device
- **ou8_Axis_Mode** – Set the device to either 'Dual-Axis' or 'Single-Axis' mode. Use the library constants INC_CFG_NO_CHANGE, INC_AXIS_MODE_DUAL, and INC_AXIS_MODE_SINGLE.
- **ou8_Pitch_Polarity** – Set the positive and negative direction of the pitch rotation. The default is CCW = positive. Use the library constants INC_CFG_NO_CHANGE, INC_POLARITY_CCW, and INC_POLARITY_CW.
- **ou8_Roll_Polarity** – Set the positive and negative direction of the roll rotation. The default is CCW = positive. Use the library constants INC_CFG_NO_CHANGE, INC_POLARITY_CCW, and INC_POLARITY_CW.
- **ou8_Pitch_and_Roll_Orientation** – Set the pitch and roll axis orientation. The default is standard orientation. Use the library constants INC_CFG_NO_CHANGE, INC_ORIENT_STANDARD, and INC_ORIENT_SWITCHED.
- **ou8_Vertical_Plane** – Manually set the vertical plane of the device. The default is the Z-Plane. Use the library constants INC_CFG_NO_CHANGE, INC_VERTICAL_PLANE_Z, …_X, …_Y, and INC_VERTICAL_PLANE_NEGATIVE_Z, …_X, …_Y.
- **ou8_Vibe_Filter** – Set the device to either 'Low Noise' or 'Noise Block' mode. The default is noise block mode. Use the library constants INC_CFG_NO_CHANGE, INC_VIBE_FILTER_LOW, and INC_VIBE_FILTER_BLOCK.
- **ou8_Sensor_Data_Transmitted** – The sensors which are currently transmitting the 'Slope Sensor Information' message. Use the library constants INC_CFG_NO_CHANGE, INC_DATA_TX_ALL, INC_DATA_TX_COMPENSATED, INC_DATA_TX_SENSOR_DATA, and INC_DATA_TX_SINGLE_SENSOR.
- **ou16_Transmit_Interval_ms** – The interval at which the inclinometer data is transmitted. The default is 10ms. Range 10ms to 2500ms in increments of 10ms.
- **of32_Accelerometer_Increment_pct** – The stored value for the percentage at which the accelerometer angle is added to the reported angle. The default is 1%. Range 0.1%-10% in increments of 0.1%.
- **opt_CAN_Msg** – Pointer to user's copy of T_x_can_msg.

**- Return -**

- **C_NO_ERR** – Function executed without error

- **C_RANGE** – Invalid Sensor source address, sample size, transmit interval, or accelerometer increment.

**- Usage -**

Execution of this function by the user is required only if they are manually transmitting CAN traffic instead of using F_INC_TX_DEV_SET. This function is executed internally by F_INC_TX_DEV_SET to encode outgoing "Device Settings Command" messages.

```
FUNCTION F_INC_TX_DEV_SET : INT
VAR_INPUT
   ou32_TX_Handle : UDINT;
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou8_Sample_Size : USINT
   oq_Zero_Sensors : BOOL;
   oq_Detect_Vertical_Plane : BOOL;
   oq_Clear_Offset_Adjustment : BOOL;
   oq_Clear_Zero_Offset : BOOL;
   ou8_Axis_Mode : USINT;
   ou8_Pitch_Polarity : USINT;
   ou8_Roll_Polarity : USINT;
   ou8_Pitch_and_Roll_Orientation : USINT;
   ou8_Vertical_Plane : USINT;
   ou8_Vibe_Filter : USINT;
   ou8_Sensor_Data_Transmitted : USINT;
   ou16_Transmit_Interval_ms : UINT;
   of32_Accelerometer_Increment_pct : REAL;
END_VAR
```

**- Inputs -**

- **ou32_TX_Handle** – Handle of the TX object for transmitting messages, created by F_INC_Init_CAN_Objects or manually by the user application.
- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **ou8_Sample_Size** – The number of measurements that are averaged together when calculating the angle. The default is 20. Range 1-20
- **oq_Zero_Sensors** – TRUE = Set the current position of the device as zero for the pitch and roll
- **oq_Detect_Vertical_Plane** – TRUE = Automatically detect and set the vertical plane of the device
- **oq_Clear_Offset_Adjustment** – TRUE = Set the user defined pitch and roll offset adjustments to zero
- **oq_Clear_Zero_Offset** – TRUE = Clear the offset that was used to zero the device
- **ou8_Axis_Mode** – Set the device to either 'Dual-Axis' or 'Single-Axis' mode. Use the library constants INC_CFG_NO_CHANGE, INC_AXIS_MODE_DUAL, and INC_AXIS_MODE_SINGLE.
- **ou8_Pitch_Polarity** – Set the positive and negative direction of the pitch rotation. The default is CCW = positive. Use the library constants INC_CFG_NO_CHANGE, INC_POLARITY_CCW, and INC_POLARITY_CW.
- **ou8_Roll_Polarity** – Set the positive and negative direction of the roll rotation. The default is CCW = positive. Use the library constants INC_CFG_NO_CHANGE, INC_POLARITY_CCW, and INC_POLARITY_CW.
- **ou8_Pitch_and_Roll_Orientation** – Set the pitch and roll axis orientation. The default is standard orientation. Use the library constants INC_CFG_NO_CHANGE, INC_ORIENT_STANDARD, and INC_ORIENT_SWITCHED.
- **ou8_Vertical_Plane** – Manually set the vertical plane of the device. The default is the Z-Plane. Use the library constants INC_CFG_NO_CHANGE, INC_VERTICAL_PLANE_Z, …_X, …_Y, and INC_VERTICAL_PLANE_NEGATIVE_Z, …_X, …_Y.
- **ou8_Vibe_Filter** – Set the device to either 'Low Noise' or 'Noise Block' mode. The default is noise block mode. Use the library constants INC_CFG_NO_CHANGE, INC_VIBE_FILTER_LOW, and INC_VIBE_FILTER_BLOCK.

- **ou8_Sensor_Data_Transmitted** – The sensors which are currently transmitting the 'Slope Sensor Information' message. Use the library constants INC_CFG_NO_CHANGE, INC_DATA_TX_ALL, INC_DATA_TX_COMPENSATED, INC_DATA_TX_SENSOR_DATA, and INC_DATA_TX_SINGLE_SENSOR.
- **ou16_Transmit_Interval_ms** – The interval at which the inclinometer data is transmitted. The default is 10ms. Range 10ms to 2500ms in increments of 10ms.
- **of32_Accelerometer_Increment_pct** – The stored value for the percentage at which the accelerometer angle is added to the reported angle. The default is 1%. Range 0.1%-10% in increments of 0.1%.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – Error in the internal encoding function, message not sent
- **C_...** – Status return of the internal transmit function block FB_x_can_obj_send_msg, message may have not been sent

**- Usage -**

This function must be called by the user to transmit the "Device Settings Command" message to the Module. Note: The Module must be in Service Mode to react to configuration command messages.

```
FUNCTION F_INC_Encode_OFFSET_ADJ : INT
VAR_INPUT
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   of32_Pitch_Offset_Adjustment_deg : REAL;
   of32_Roll_Offset_Adjustment_deg : REAL;
   opt_CAN_Msg : POINTER TO T_x_can_msg;
END_VAR
```

**- Inputs -**

- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **of32_Pitch_Offset_Adjustment_deg** – The adjustment to the pitch offset. Range -65.534 to 65.534 degrees in 0.002 degree increments.
- **of32_Roll_Offset_Adjustment_deg** – The adjustment to the roll offset. Range -65.534 to 65.534 degrees in 0.002 degree increments.
- **opt_CAN_Msg** – Pointer to user's copy of T_x_can_msg.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_RANGE** – Invalid Sensor source address

**- Usage -**

Execution of this function by the user is required only if they are manually transmitting CAN traffic instead of using F_INC_TX_OFFSET_ADJ. This function is executed internally by F_INC_TX_OFFSET_ADJ to encode outgoing "Sensor Offset Adjustment Command" messages.

```
FUNCTION F_INC_TX_OFFSET_ADJ : INT
VAR_INPUT
   ou32_TX_Handle : UDINT;
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   of32_Pitch_Offset_Adjustment_deg : REAL;
   of32_Roll_Offset_Adjustment_deg : REAL;
END_VAR
```

**- Inputs -**

- **ou32_TX_Handle** – Handle of the TX object for transmitting messages, created by F_INC_Init_CAN_Objects or manually by the user application.
- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **of32_Pitch_Offset_Adjustment_deg** – The adjustment to the pitch offset. Range -65.534 to 65.534 degrees in 0.002 degree increments.
- **of32_Roll_Offset_Adjustment_deg** – The adjustment to the roll offset. Range -65.534 to 65.534 degrees in 0.002 degree increments.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – Error in the internal encoding function, message not sent
- **C_...** – Status return of the internal transmit function block FB_x_can_obj_send_msg, message may have not been sent

**- Usage -**

This function must be called by the user to transmit the "Sensor Offset Adjustment Command" message to one of the Module's sensor channels. Note: The Module must be in Service Mode to react to configuration command messages.

```
FUNCTION F_INC_Encode_REQUEST : INT
VAR_INPUT
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   oq_Revision_Request : BOOL;
   oq_Stored_Settings_Request : BOOL;
   oq_Stored_Offsets_Request : BOOL;
   oq_Raw_Accel_Request : BOOL;
   oq_Raw_Gyro_Request : BOOL;
   oq_Temp_Request : BOOL;
   oq_Acc_Angle_Request : BOOL;
   oq_Calibrated_Accel_Request : BOOL;
   opt_CAN_Msg : POINTER TO T_x_can_msg;
END_VAR
```

**- Inputs -**

- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **oq_Revision_Request** – Set the device to transmit the "Revision and Serial Number" message once
- **oq_Stored_Settings_Request** – Set the device to transmit the "Stored Settings Data" message once
- **oq_Stored_Offsets_Request** – Set the device to transmit the "Stored Offsets Data" message once
- **oq_Raw_Accel_Request** – Set the device to transmit the "Raw Acceleration Data" message once
- **oq_Raw_Gyro_Request** – Set the device to transmit the "Raw Gyroscope Data" message once
- **oq_Temp_Request** – Set the device to transmit the "Temperature Data" message once
- **oq_Acc_Angle_Request** – Set the device to transmit the "Accelerometer Angle Data" message once
- **oq_Calibrated_Accel_Request** – Set the device to transmit the "Calibrated Acceleration Data" message once
- **opt_CAN_Msg** – Pointer to user's copy of T_x_can_msg.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_RANGE** – Invalid Sensor source address

**- Usage -**

Execution of this function by the user is required only if they are manually transmitting CAN traffic instead of using F_INC_TX_REQUEST. This function is executed internally by F_INC_TX_REQUEST to encode outgoing "Data Request Command" messages. Note: The compensated sensor channel will only respond to requests for "Revision and Serial Number" and "Stored Settings Data". Other data is unavailable on this channel.

```
FUNCTION F_INC_TX_REQUEST : INT
VAR_INPUT
   ou32_TX_Handle : UDINT;
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   oq_Revision_Request : BOOL;
   oq_Stored_Settings_Request : BOOL;
   oq_Stored_Offsets_Request : BOOL;
   oq_Raw_Accel_Request : BOOL;
   oq_Raw_Gyro_Request : BOOL;
   oq_Temp_Request : BOOL;
   oq_Acc_Angle_Request : BOOL;
   oq_Calibrated_Accel_Request : BOOL;
END_VAR
```

**- Inputs -**

- **ou32_TX_Handle** – Handle of the TX object for transmitting messages, created by F_INC_Init_CAN_Objects or manually by the user application.
- **ou8_Own_SA** – The source address of the ECU.

- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **oq_Revision_Request** – Set the device to transmit the "Revision and Serial Number" message once
- **oq_Stored_Settings_Request** – Set the device to transmit the "Stored Settings Data" message once
- **oq_Stored_Offsets_Request** – Set the device to transmit the "Stored Offsets Data" message once
- **oq_Raw_Accel_Request** – Set the device to transmit the "Raw Acceleration Data" message once
- **oq_Raw_Gyro_Request** – Set the device to transmit the "Raw Gyroscope Data" message once
- **oq_Temp_Request** – Set the device to transmit the "Temperature Data" message once
- **oq_Acc_Angle_Request** – Set the device to transmit the "Accelerometer Angle Data" message once
- **oq_Calibrated_Accel_Request** – Set the device to transmit the "Calibrated Acceleration Data" message once

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – Error in the internal encoding function, message not sent
- **C_...** – Status return of the internal transmit function block FB_x_can_obj_send_msg, message may have not been sent

**- Usage -**

This function must be called by the user to transmit the "Data Request Command" message to one of the Module's sensor channels. Note: The compensated sensor channel will only respond to requests for "Revision and Serial Number" and "Stored Settings Data". Other data is unavailable on this channel.

```
FUNCTION F_INC_Encode_SERVICE : INT
VAR_INPUT
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou32_Service_Mode_Enable : UDINT;
   opt_CAN_Msg : POINTER TO T_x_can_msg;
END_VAR
```

**- Inputs -**

- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **ou32_Service_Mode_Enable**– The passcode needed to place the unit into service mode. Use library constants INC_SERVICE_MODE_ENABLE and INC_SERVICE_MODE_DISABLE.
- **opt_CAN_Msg** – Pointer to user's copy of T_x_can_msg.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_RANGE** – Invalid Sensor source address

**- Usage -**

Execution of this function by the user is required only if they are manually transmitting CAN traffic instead of using F_INC_TX_SERVICE. This function is executed internally by F_INC_TX_SERVICE to encode outgoing "Service Mode Enable Command" messages.

```
FUNCTION F_INC_TX_SERVICE : INT
VAR_INPUT
   ou32_TX_Handle : UDINT;
   ou8_Own_SA : USINT;
   ou8_Sensor_SA : USINT;
   ou32_Service_Mode_Enable : UDINT;
END_VAR
```

**- Inputs -**

- **ou32_TX_Handle** – Handle of the TX object for transmitting messages, created by F_INC_Init_CAN_Objects or manually by the user application.
- **ou8_Own_SA** – The source address of the ECU.
- **ou8_Sensor_SA** – The source address of the Module's sensor channel. Range 0-254
- **ou32_Service_Mode_Enable**– The passcode needed to place the unit into service mode. Use library constants INC_SERVICE_MODE_ENABLE and INC_SERVICE_MODE_DISABLE.

**- Return -**

- **C_NO_ERR** – Function executed without error
- **C_NOACT** – Error in the internal encoding function, message not sent
- **C_...** – Status return of the internal transmit function block FB_x_can_obj_send_msg, message may have not been sent

**- Usage -**

This function must be called by the user to transmit the "Service Mode Enable Command" message to one of the Module's sensor channels. This message must be used to enable service mode for each sensor channel before any configuration messages will be accepted by the Module.

# 4 Software Release History

| Release | Date MM/DD/YYYY | Description |
|---|---|---|
| **V2.0r0** | 03/11/2020 | Initial release. |

# 5 Known Issues

| Issue # | Date MM/DD/YYYY | Description | Version | Status |
|---|---|---|---|---|
| *No known issues* | | | | |

# 6 Document History

| Rev | Author | Date MM/DD/YYYY | Change Made |
|---|---|---|---|
| **2** | STW | 03/11/2020 | Update for library v2 |
| **1** | STW | xx/xx/2016 | Document created. |