Allora International

Adaptive Touch Technology
OEM USER GUIDE

Rev 1.00

Purpose

This document is provided to the OEM and CANopen bus integrator. It describes the interface and usage for the Allora executable library functions embedded in the PCS ER+ sensor. In-depth knowledge of the CANopen protocol is assumed. *This information is not required for installation and application of Allora-provided sensor controllers as these functions are built-in.*

A quick-start tutorial is provided at the end of this document as a means to evaluate the function of the sensor. This also provides a common platform to investigate service related issues.

Terms, Abbreviations, and Definitions

Explanation of terms as they apply to the PCS ER+ system.

Term	Meaning
Bit Rate	The operating speed of the CANbus. Set at 500kHz by default
Command	The execution of a library function when initiated by a SDO write to Object Dictionary location 0x3005.01. A function may also be executed by a TxPDO write to 0x200+Node_Id.
CW, CCW	Direction of rotation (clockwise/counterclockwise) of the sensor as viewed from the wand end
Detect	The process of sweeping the wand to determine the presence of a previously-learned target
DLC	Data length code. Length of data carried in the CAN bus message
Node ID	The unique address of a specific sensor on the bus. Legitimate values are defined by the CANopen standard
OD	Object Dictionary (as defined by the CANopen standard)
Speed/Velocity	The rate of approach of the wand to the target
Target	The object to be detected/verified by the system
Target Position	The angular position of the target
Teach	The process of sweeping the sensor wand to learn the location of a target prior to executing the detect function
Tolerance	The allowable error in target position (positive and negative)
Torque	The amount of force (drive current) applied to the motor to move the wand toward the target position
Wand	The 'arm' attached to the motor to contact the target

System Overview

The Allora Adaptive Touch Technology system integrates a function library and data management required to implement a smart device dedicated to the role of target detection and fault reporting. There are two major pieces of the system firmware: **settable parameter variables** and **executable library functions**

The system is capable of testing targets in both directions so there are separate parameters and functions for clockwise and counter-clockwise rotations (CW/CCW) rotations. Note that rotation directions are defined as viewed from the wand end of the sensor.

The executable library contains a set of pre-defined functions specific to system configuration, target detection, and status reporting. They are all accessed through the standard CANopen Object Dictionary functions (SDO protocol) at address 0x3005, sub-index 0x01 (0x3005.01).

The settable parameters are also addressed through the 0x3005 address at various sub-indices. These control values for sweep speed, force, and tolerance windows among others.

The bus master (PLC, PC, etc.) executes library commands utilizing the standard CANopen SDO protocol. Status and test result indication can be implemented using SDO or PDO methods. See the quick start tutorial at the end of this document for examples of interfacing controlling the system with common CAN bus tools.

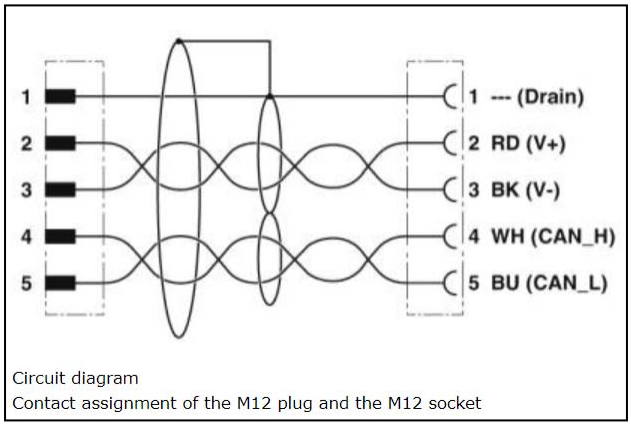
The sensors are fully CANopen compliant and may be used with other CANopen devices on the network.

Hardware Configuration

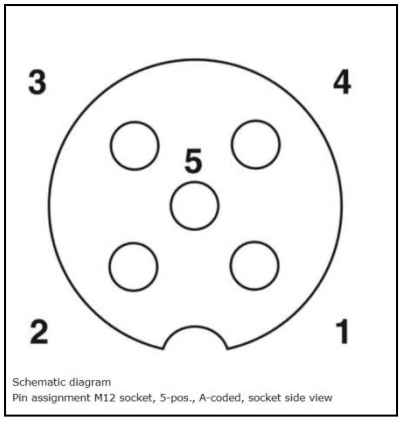
Bit Rate: The default bit rate is 500kHz. This can be changed using normal CANopen methods

Power Supply: The system is powered from a dedicated +24V (+- 10%) supply. The current requirement is dependent on the number of nodes (sensors) on the bus. Each sensor can briefly draw as much as 2A (about 200mA when idle).

Connector info: The connector is a circular, 5 pin, ‘A’ coded, M12. Color coding and signal definitions are shown below



Circuit diagram
Contact assignment of the M12 plug and the M12 socket



Cables: All cabling follows the CANopen/Devicenet conventions. For reliability, cables should be compliant to those standards.

Node Id Management

Each device on the network must be assigned a unique node ID. Duplicate Node IDs will cause the network to fail.

New sensors are supplied from the factory with Node ID = 127.

One strategy for management is to reserve Node ID #127 as a configuration address. If a factory-new sensor is to be added to the network, it can be inserted with the unique id as 127, then, changed to another number using the Function Library commands. Also, if a sensor is to be removed from a network, it can first be assigned Node ID #127 (or set to any unused ID), removed, then relocated to another network and inserted as above. Thus, any sensor not in a network is maintained as Node ID #127.

Things get more complicated when the Node ID of a sensor is unknown. In this case, a single-node network must be created either stand-alone or by unplugging the other nodes in the network. When the unknown device is powered, it will report the CANopen boot message (0x700 + Node_Id). The node id can then be changes using the built-in methods of this function library.

Because this is an open standard there are other methods and tools to perform this task

Power-up and Initialization

On every power up, per the CANopen protocol, each unit will transmit a boot up message of 0x700 + Node ID. This can identify nodes on the network.

At the first power-on of a new unit it stays in a zero velocity mode. The wand can be moved manually to establish a home position. Then a Set_Home_Position() command is executed. This position is stored as the operational home and is stored in non-volatile memory.

On subsequent power-ups, the sensor will return to its predefined home position. This will require some time interval, during which the controller must not send additional commands. A 'READY' transmission will indicated that the sensor is able to receive commands. See the Power-up example in the Example Transactions section.

Configuration parameters are stored in the sensor and remain through power cycle. The values can be read from the sensor for display/modification.

Object Dictionary location 0x3005

Object dictionary location 0x3005 implements 26 (0x1a) sub-index locations; all are 32 bit values. Communication, configuration, and control of the sensor take place through this set of registers.

The table below summarizes the content and function for each location. Note that 'Reserved' locations should not be accessed.

Access properties:
R – read only
RW – read/write
W – write only

Sub Index	Symbol	Access	Default value	Function	Description
0x00		R	26 (0x19)	Constant	Number of sub-index locations
Command execution and status reporting					
0x01	a	RW	0	Library function to execute	SDO write function number
0x02	b	RW	0	Function execution status	Returns 0 for OK or 255 (0xFF) for function FAIL Must be cleared (write 0) on FAIL
0x03	c	RW	0	Function execution result	Command specific Must be cleared (write 0) after a non-zero read
System variables					
0x04	d	R	0	Motor Encoder Position Offset	Last known internal encoder position
0x05	e	RW	0	Job Cycle Counter	Counts the number of 'detect' cycles. This is managed by the end user. It can be reset to zero or other value by a write to this variable. It is volatile and will be lost on power cycle.

0x06	f	R	0	Life Cycle Counter	Number of 'detect' cycle for life of unit. Non-resettable. This value is saved to permanent memory at 30 minute intervals
0x07	g	RW	0	Job Time Counter	Time (in 10 minute increments). This is managed by the end user. It can be reset to zero or other value by a write to this variable. It is volatile and will be lost on power cycle.
0x08	h	R	0	Life Time Timer	Sensor powered on time. Timer increments and updates new value to permanent memory at 30 minute intervals.
0x09	i	N/A	0	N/A	Reserved
0x0a	j	N/A	x	N/A	Reserved
0x0b	k	RW	0	CW Target Position	Target position for clockwise detect.
0x0c	l	RW	50	CW Detect Velocity	Speed of approach to target for detect. CW direction
0x0d	m	RW	200	CW Detect Force	Torque value for approach to target for detect. CW direction
0x0e	n	RW	29	CW Detect Tolerance	The allowable error for detection of the target. CW direction
0x0f	o	RW	0	CCW Target Position	Target position for counter-clockwise detect.
0x10	p	RW	50	CCW Detect Velocity	Speed of approach to target for detect. CCW direction
0x11	q	RW	200	CCW Detect Force	Torque value for approach to target for detect. CCW direction
0x12	r	RW	29	CCW Detect Tolerance	The allowable error for detection of the target. CCW direction
0x13	s	N/A	0	N/A	Reserved
0x14	t	R	0	Watchdog Timer Flag	Indicates that function took too long to execute
0x15	u	N/A	0	N/A	Reserved
0x16	v	R	0	Cycle Timer Flag	
0x17	w	R	0	Firmware Revision	Firmware revision in two low-order bytes. Byte[0] is minor rev. Byte[1] is major revision
0x18	x	R	0	Torque Detect Delta	Used internally
0x19	y	R	0	First Power On	Used internally to determine

					power-on behavior
0x1a	z	R	0	Tool Monitor	Return of status combined to single word. B0 (LSB) = Execution result B1 = Function number/status B2 = Function number B3 (MSB) = Status (READY bit)

Detect Parameters

The detect functions are controlled by these three variables (Velocity, Force, Tolerance) and are adjusted to affect system behavior. These variables can be set independently for each direction of travel. Also, actual performance is dependent on numerous external factors such as:

- Orientation (is the sensor working with or against gravity)
- Seal condition (some coolants and lubricants may cause stickiness in the system)
- Bouncing (severe wand bounce can cause erratic readings and false alarms
- Wand flex (long, thin wands can flex significantly when driven at high torque and rate

Detect Velocity (CW/CCW): The valid range for this parameter is 1 to 1000. In practice, the usable range is from about 10 for a delicate target to about 300 for a more robust stop.

Detect Force (CW/CCW): The valid range for this parameter is 1 to 1000. The usable range for this parameter is from about 100 to 500 depending on circumstances as listed above.

Detect Tolerance (CW/CCW): Range is 1 to 4095 and represents raw encoder counts. A setting of 29 counts represents approximately 2.5 degrees. The practical range for this parameter is from about 30 to 100 (2.5 to 9 degrees) depending on circumstances as listed above.

Function Execution Behavior

A library function command executes immediately when the function code is written to 0x3005, sub-index 01 (0x3005.01). The write can be initiated by SDO write command or by a PDO transaction sent to RXPDO_1 (0x200 + Node_ID).

Command completion can be detected either through SDO polling of the 3005.02 and 3005.03 registers or by reception of the READY bit which is sent from the sensor asynchronously via TXPDO_1 (0x180 + Node_ID).

While the function is in progress (some functions may take hundreds of milliseconds to complete), reads from location 0x3005.02 will return the function number as written.

When the command completes, reads from 0x3005.02 will return a 0 if the command completes normally, or returns 255 (0xff) on failure. **Note that this is a command failure - which is distinct from a**

test result fault. For example, in the case where a 'Detect' testing function is executed, the command may execute properly (returning a '0'), but the result of the test may indicate an under-travel (obstruction) or over-travel (broken tool), etc. The test result code is then read from 0x3005.03

A test result is always read from 0x3005.03 and will return a specific result code.

Error and result codes must be explicitly cleared (by writing a 0) prior to initiation of a new command.

Function Library (Commands)

See details of function behavior below.

Command	Name	Description
0 (0x00)	Cancel_Command()	Used by Jog_CW(), Jog_CCW() to terminate the command
1 (0x01)	Hold_Zero_Speed()	Sets the motor to 'zero speed' mode. The wand can be rotated manually to establish a home position or target positions as needed.
2 (0x02)	Hold_Position()	Holds current position
3 (0x03)	Step_CW()	Performs a relative movement of 10 degrees (116 counts) in clockwise direction
4 (0x04)	Step CCW()	Performs a relative movement of 10 degrees (116 counts) in counter-clockwise direction
5 (0x05)	Set_Home_Position()	Assigns (learns) the current wand position as the 'home' position. Immediately saved in non-volatile memory.
6 (0x06)	Go_Home()	Enables position mode (if not already) and returns the wand to the currently-defined home position
7 (0x07)	Teach_CW()	Establish a target position in the CW direction. Wand will move until an obstruction is detected, then return to the home position. Immediately saved in non-volatile memory.
8 (0x08)	Teach_CCW()	Establish a target position in the CW direction. Wand will move until an obstruction is detected, then return to the home position. Immediately saved in non-volatile memory.
9 (0x09)	Load_Detect_Params_CW()	Load clockwise detection parameters (velocity, force, and tolerance) from memory to operational space
10 (0x0a)	Load_Detect_Params_CCW()	Load counter-clockwise detection parameters (velocity, force, and tolerance) from memory to operational space
11 (0x0b)	Save_Detect_Params_CW()	Save clockwise detection parameters (velocity, force, and tolerance) to non-volatile memory
12 (0x0c)	Save_Detect_Params_CCW()	Save counter-clockwise detection parameters (velocity, force, and tolerance) to non-volatile memory
13 (0x0d)	Detect_CW()	Initiate a sequence to detect the presence (or absence) of a previously learned target position. The wand will advance in the CW direction at a preset high rate of speed then slow as it approaches the target.

		It will then proceed until the target is detected by stalling the advance, or the wand position exceeds the position by the 'detect tolerance parameter' setting. The wand will then return to home position. The stall condition is controlled by two factors, 'detect velocity' and 'detect torque'. These are individually adjustable; lower values for smaller (more delicate) targets. On proper completion of the detect cycle, the command response is '0' (same as other commands). This command will also provide a test result indicating the state of the target. These results are: 0 – OK, the target was properly detected within tolerance 1 – Over-travel, the wand traveled beyond the target position and the tolerance range. 2 – Under-travel, the target position was blocked by an obstruction and the wand never attained the target position within the tolerance range.
14 (0x0e)	Detect_CCW()	See Detect_CW() above. Operation is the same in the opposite direction.
15 (0x0f)	Teach_Manual_CW()	The current wand position is defined as the CW target position. In conjunction with a previously-executed 'Hold_Zero_Speed()' command, this allows the wand to be manually moved to the target position. Or end of motion for free space monitoring.
16 (0x10)	Teach_Manual_CCW()	See Teach_Manual_CW() above. This defines the target in the CCW direction.
17 (0x11)	Jog_CW()	Starts the wand moving (at slow speed) in the CW direction. It will continue in motion until the command is cancelled with a Cancel_Command() or until a 6 second timeout (one rotation time) expires
18 (0x12)	Jog_CCW	See Jog_CW() command above. Moves in the CCW direction.
19 (0x13)	Factory_Reset()	Resets sensor to original factory settings (not lifetime count or cycle). Node ID is not changed. Then a NMT Stop command, then NMT Reset or power cycle is required.
20 (0x14)	ER_Parameter_Reset()	Resets Job Timer, Job Counter, default force, speed, tolerance, home position, and tool target positions (CW/CCW).

Example Transactions and Function Execution

These examples show both the polling method and the READY bit behavior in TxPDO1. Either (or both) may be used in practice. The polling method first polls the function status register (3005.02). It will return a '0' if OK, a 255(0xff) on failure. After 3005.02 returns OK, its result, 3005.03 is read.

Note the following about the TxPDO1 messaging and the READY bit. This applies to all functions.

- Multiple transmissions may occur while the function is in progress
- Error conditions may be presented when READY bit is clear (not asserted). May be ignored.
- There will be two TxPDO1 transmission with READY bit asserted at completion of the function. The second transmission is redundant and may be ignored.
- More info on the TxPDO1 message format and the READY bit is available in 'Appendix A'.

All of these examples show both periodic SDO polling and asynchronous PDO transmissions. The SDO polling is not required. After starting a function, the host can simply wait for the PDO response and 'Ready' bit. Periodic SDO polling request/responses are highlighted in blue text for visibility; they are optional. Other behaviors of interest may be highlighted in yellow.

Power-up

A power-up transaction is shown. After the 0x700 Boot message is received, allow 3 seconds for the sensor to initialize. After that, The NMT Operational message is sent. The sensor then moves to its home position. Note that this move to home position may take some time. In this example, it takes over 7 seconds to reposition after the NMT Operational command is issued.

The READY bit is then asserted and the sensor is ready to process commands. In this example, the script revision is requested after the READY bit (sent with TxPDO1) is asserted. The second TxPDO1 message is redundant and may be ignored.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000000271 0701 01 00                             BOOT (Node Id = 1)
Note 3 second delay - required
0000003290 0000 02 01 01                        NMT Operational
0000003291 0181 06 27 00 00 00 00 01              TxPDO1 z = 1, 0, 0, 0   READY bit
0000003300 0601 08 40 05 30 17 00 00 00 00   REQ
0000003301 0581 08 43 05 30 17 15 00 00 00        (w) Script Rev = 0.21
```

Function #1, Hold_Zero_Speed()

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000313570 0601 08 23 05 30 01 01 00 00 00   CMD Hold_Zero_Speed()
0000313571 0581 08 60 05 30 01 00 00 00 00        ACK
0000313580 0181 06 27 54 00 01 01 00              TxPDO1 z = 0, 1, 1, 0
0000313586 0181 06 27 14 00 00 01 00              TxPDO1 z = 0, 1, 0, 0
0000313592 0181 06 27 54 00 00 01 01              TxPDO1 z = 1, 1, 0, 0   READY bit
0000313599 0181 06 27 14 00 00 01 01              TxPDO1 z = 1, 1, 0, 0   redundant message
0000313670 0601 08 40 05 30 02 00 00 00 00   REQ
0000313671 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000313671 0601 08 40 05 30 03 00 00 00 00   REQ
0000313673 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #2, Hold_Position()

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0002329950 0601 08 23 05 30 01 02 00 00 00   CMD Hold_Position()
0002329951 0581 08 60 05 30 01 00 00 00 00        ACK
0002329959 0181 06 27 44 00 02 02 00              TxPDO1 z = 0, 2, 2, 0
0002329963 0181 06 27 04 00 00 02 00              TxPDO1 z = 0, 2, 0, 0
0002329969 0181 06 27 44 00 00 02 01              TxPDO1 z = 1, 2, 0, 0
0002329970 0181 06 27 04 00 00 02 01              TxPDO1 z = 1, 2, 0, 0
0002330050 0601 08 40 05 30 02 00 00 00 00   REQ
0002330051 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0002330051 0601 08 40 05 30 03 00 00 00 00   REQ
0002330053 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #3, #4 Step_CW(), Step_CCW()

These functions operate the same. Only difference is the direction of travel. Note that due to the execution time of this function multiple polling events are captured in this example.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0002513850 0601 08 23 05 30 01 03 00 00 00   CMD Step_CW()
0002513851 0581 08 60 05 30 01 00 00 00 00        ACK
0002513855 0181 06 27 44 00 03 03 00              TxPDO1 z = 0, 3, 3, 0
0002513858 0181 06 27 54 00 03 03 00              TxPDO1 z = 0, 3, 3, 0
0002513859 0181 06 27 44 00 03 03 00              TxPDO1 z = 0, 3, 3, 0
0002513860 0181 06 27 40 00 03 03 00              TxPDO1 z = 0, 3, 3, 0
0002513950 0601 08 40 05 30 02 00 00 00 00   REQ
0002513951 0581 08 43 05 30 02 03 00 00 00        (b) Funct Status = 3
0002514050 0601 08 40 05 30 02 00 00 00 00   REQ
0002514051 0581 08 43 05 30 02 03 00 00 00        (b) Funct Status = 3
0002514150 0601 08 40 05 30 02 00 00 00 00   REQ
0002514151 0581 08 43 05 30 02 03 00 00 00        (b) Funct Status = 3
0002514250 0601 08 40 05 30 02 00 00 00 00   REQ
0002514251 0581 08 43 05 30 02 03 00 00 00        (b) Funct Status = 3
0002514350 0601 08 40 05 30 02 00 00 00 00   REQ
0002514351 0581 08 43 05 30 02 03 00 00 00        (b) Funct Status = 3
0002514359 0181 06 27 44 00 03 03 00              TxPDO1 z = 0, 3, 3, 0
0002514364 0181 06 27 04 00 00 03 00              TxPDO1 z = 0, 3, 0, 0
0002514370 0181 06 27 44 00 00 03 01              TxPDO1 z = 1, 3, 0, 0
0002514376 0181 06 27 04 00 00 03 01              TxPDO1 z = 1, 3, 0, 0
0002514450 0601 08 40 05 30 02 00 00 00 00   REQ
0002514451 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0002514451 0601 08 40 05 30 03 00 00 00 00   REQ
0002514453 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #5, Set_Home_Position()

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0002685350 0601 08 23 05 30 01 05 00 00 00   CMD Set_Home()
0002685351 0581 08 60 05 30 01 00 00 00 00        ACK
0002685358 0181 06 27 00 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0002685359 0181 06 27 44 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0002685360 0181 06 27 54 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0002685364 0181 06 27 44 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0002685365 0181 06 27 40 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0002685366 0181 06 27 50 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0002685367 0181 06 27 54 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0002685378 0181 06 27 14 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0002685383 0181 06 27 54 00 00 05 01              TxPDO1 z = 1, 5, 0, 0
0002685392 0181 06 27 14 00 00 05 01              TxPDO1 z = 1, 5, 0, 0
0002685450 0601 08 40 05 30 02 00 00 00 00   REQ
0002685451 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0002685451 0601 08 40 05 30 03 00 00 00 00   REQ
0002685453 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #6, Go_Home()

The time for this function to execute will depend on the distance to be travelled to the home position.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0002818980 0601 08 23 05 30 01 06 00 00 00   CMD Go_Home()
0002818981 0581 08 60 05 30 01 00 00 00 00       ACK
0002818987 0181 06 27 00 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002818988 0181 06 27 40 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002818990 0181 06 27 50 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002818991 0181 06 27 40 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002819080 0601 08 40 05 30 02 00 00 00 00   REQ
0002819081 0581 08 43 05 30 02 06 00 00 00       (b) Funct Status = 6
0002819180 0601 08 40 05 30 02 00 00 00 00   REQ
0002819181 0581 08 43 05 30 02 06 00 00 00       (b) Funct Status = 6
0002819280 0601 08 40 05 30 02 00 00 00 00   REQ
0002819281 0581 08 43 05 30 02 06 00 00 00       (b) Funct Status = 6
0002819380 0601 08 40 05 30 02 00 00 00 00   REQ
0002819381 0581 08 43 05 30 02 06 00 00 00       (b) Funct Status = 6
0002819480 0601 08 40 05 30 02 00 00 00 00   REQ
0002819481 0581 08 43 05 30 02 06 00 00 00       (b) Funct Status = 6
0002819498 0181 06 27 44 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002819503 0181 06 27 04 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0002819509 0181 06 27 44 00 00 06 01             TxPDO1 z = 1, 6, 0, 0
0002819517 0181 06 27 04 00 00 06 01             TxPDO1 z = 1, 6, 0, 0
0002819580 0601 08 40 05 30 02 00 00 00 00   REQ
0002819581 0581 08 43 05 30 02 00 00 00 00       (b) Funct Status = 0 - OK
0002819581 0601 08 40 05 30 03 00 00 00 00   REQ
0002819583 0581 08 43 05 30 03 00 00 00 00       (c) Funct Result = 0 - OK
```

Function #7, #8, Teach_CW(), Teach_CCW()

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0002918580 0601 08 23 05 30 01 07 00 00 00   CMD Teach_CW()
0002918581 0581 08 60 05 30 01 00 00 00 00       ACK
0002918588 0181 06 27 44 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002918632 0181 06 27 54 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002918633 0181 06 27 50 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002918634 0181 06 27 40 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002918680 0601 08 40 05 30 02 00 00 00 00   REQ
0002918681 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002918780 0601 08 40 05 30 02 00 00 00 00   REQ
0002918781 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002918880 0601 08 40 05 30 02 00 00 00 00   REQ
0002918881 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002918980 0601 08 40 05 30 02 00 00 00 00   REQ
0002918981 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002919080 0601 08 40 05 30 02 00 00 00 00   REQ
0002919081 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002919180 0601 08 40 05 30 02 00 00 00 00   REQ
0002919181 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002919280 0601 08 40 05 30 02 00 00 00 00   REQ
0002919281 0581 08 43 05 30 02 07 00 00 00       (b) Funct Status = 7
0002919283 0181 06 27 50 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919285 0181 06 27 54 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919286 0181 06 27 40 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919289 0181 06 27 50 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919290 0181 06 27 40 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919380 0601 08 40 05 30 02 00 00 00 00   REQ
0002919381 0581 08 43 05 30 02 00 00 00 00       (b) Funct Status = 0 - OK
0002919381 0601 08 40 05 30 03 00 00 00 00   REQ
0002919383 0581 08 43 05 30 03 00 00 00 00       (c) Funct Result = 0 - OK
0002919732 0181 06 27 44 00 07 07 00             TxPDO1 z = 0, 7, 7, 0
0002919738 0181 06 27 04 00 00 07 00             TxPDO1 z = 0, 7, 0, 0
0002919743 0181 06 27 44 00 00 07 01             TxPDO1 z = 1, 7, 0, 0
0002919751 0181 06 27 04 00 00 07 01             TxPDO1 z = 1, 7, 0, 0
```

Function #9, #10, Load_Detect_Params_CW(),Load_Detect_Params_CCW()

This illustrates reloading both the CW and CCW detect parameters (velocity, force, tolerance) from non-volatile memory for use on subsequent detect cycles.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003934290 0601 08 23 05 30 01 09 00 00 00   CMD Load_CW()
0003934291 0581 08 60 05 30 01 00 00 00 00        ACK
0003934296 0181 06 27 44 00 09 09 00              TxPDO1 z = 0, 9, 9, 0
0003934301 0181 06 27 04 00 00 09 01              TxPDO1 z = 1, 9, 0, 0
0003934390 0601 08 40 05 30 02 00 00 00 00   REQ
0003934391 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003934391 0601 08 40 05 30 03 00 00 00 00   REQ
0003934393 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 – OK

0003934400 0601 08 23 05 30 01 0A 00 00 00   CMD Load_CCW()
0003934401 0581 08 60 05 30 01 00 00 00 00        ACK
0003934407 0181 06 27 44 00 0A 0A 00              TxPDO1 z = 0, 10, 10, 0
0003934412 0181 06 27 04 00 00 0A 01              TxPDO1 z = 1, 10, 0, 0
0003934500 0601 08 40 05 30 02 00 00 00 00   REQ
0003934501 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003934501 0601 08 40 05 30 03 00 00 00 00   REQ
0003934503 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 – OK
```

Function #11, #12, Save_Detect_Params_CW(),Save_Detect_Params_CCW()

This illustrates saving both the CW and CCW detect parameters (velocity, force, tolerance) to non-volatile memory.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0004007420 0601 08 23 05 30 01 0B 00 00 00   CMD Save_CW()
0004007421 0581 08 60 05 30 01 00 00 00 00        ACK
0004007429 0181 06 27 44 00 0B 0B 00              TxPDO1 z = 0, 11, 11, 0
0004007441 0181 06 27 04 00 00 0B 01              TxPDO1 z = 1, 11, 0, 0
0004007520 0601 08 40 05 30 02 00 00 00 00   REQ
0004007521 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0004007521 0601 08 40 05 30 03 00 00 00 00   REQ
0004007523 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 – OK

0004007530 0601 08 23 05 30 01 0C 00 00 00   CMD Save_CCW()
0004007531 0581 08 60 05 30 01 00 00 00 00        ACK
0004007537 0181 06 27 44 00 0C 0C 00              TxPDO1 z = 0, 12, 12, 0
0004007549 0181 06 27 04 00 00 0C 01              TxPDO1 z = 1, 12, 0, 0
0004007630 0601 08 40 05 30 02 00 00 00 00   REQ
0004007631 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0004007631 0601 08 40 05 30 03 00 00 00 00   REQ
0004007632 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #13, #14, Detect_CW(),Detect_CCW()

Note that in this example, sweep parameters are sent prior to the Detect command. This is optional. Sweep parameters can be downloaded once and stored. They can then be used for subsequent commands (after executing “Load_Detect_Params_xxx”). This example illustrates an under travel fault condition in the function result register. It also shows the behavior of the asynchronous Ready bit of TxPDO1

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0004130740 0601 08 23 05 30 01 0D 00 00 00   CMD Detect_CW()
0004130741 0581 08 60 05 30 01 00 00 00 00        ACK
0004130746 0181 06 27 44 00 0D 0D 00              TxPDO1 z = 0, 13, 13, 0
0004130748 0181 06 27 50 00 0D 0D 00              TxPDO1 z = 0, 13, 13, 0
0004130749 0181 06 27 40 00 0D 0D 00              TxPDO1 z = 0, 13, 13, 0
```

```
0004130840 0601 08 40 05 30 02 00 00 00 00   REQ
0004130841 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004130934 0181 06 27 50 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004130935 0181 06 27 54 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004130936 0181 06 27 40 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004130938 0181 06 27 50 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004130939 0181 06 27 40 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004130940 0601 08 40 05 30 02 00 00 00 00   REQ
0004130941 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004131040 0601 08 40 05 30 02 00 00 00 00   REQ
0004131041 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004131140 0601 08 40 05 30 02 00 00 00 00   REQ
0004131141 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004131240 0601 08 40 05 30 02 00 00 00 00   REQ
0004131241 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004131340 0601 08 40 05 30 02 00 00 00 00   REQ
0004131341 0581 08 43 05 30 02 0D 00 00 00        (b) Funct Status = 13
0004131405 0181 06 27 44 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004131410 0181 06 27 04 02 0D 0D 00              TxPDO1 z = 0, 13, 13, 2
0004131415 0181 06 27 44 02 00 0D 01              TxPDO1 z = 1, 13, 0, 2
0004131423 0181 06 27 04 02 00 0D 01              TxPDO1 z = 1, 13, 0, 2
0004131440 0601 08 40 05 30 02 00 00 00 00   REQ
0004131441 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0004131441 0601 08 40 05 30 03 00 00 00 00   REQ
0004131443 0581 08 43 05 30 03 02 00 00 00        (c) Funct Result = 2 - UNDER

0000122540 0601 08 23 05 30 0C 64 00 00 00   SET CW Velocity = 100
0000122541 0581 08 60 05 30 0C 00 00 00 00        ACK
0000122550 0601 08 23 05 30 0D 5E 01 00 00   SET CW Torque = 350
0000122551 0581 08 60 05 30 0D 00 00 00 00        ACK
0000122560 0601 08 23 05 30 0E 58 00 00 00   SET CW Tolerance = 88
0000122561 0581 08 60 05 30 0E 00 00 00 00        ACK
```

Function #15, #16, Teach_Manual_CW(), Teach_Manual_CCW()

The current wand position is captured as the target for the respective direction of travel.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003337520 0601 08 23 05 30 01 0F 00 00 00   CMD Teach_Manual_CW()
0003337521 0581 08 60 05 30 01 00 00 00 00        ACK
0003337528 0181 06 27 54 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0003337534 0181 06 27 14 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0003337620 0601 08 40 05 30 02 00 00 00 00   REQ
0003337621 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003337621 0601 08 40 05 30 03 00 00 00 00   REQ
0003337623 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #17, #18, Jog_CW(), Jog_CCW()

As noted above, the Jog command will place the sensor in motion. It will remain in motion until terminated with a Cancel_Command() or it times out after 10 seconds. In either case, the READY bit is asserted on termination.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0005249370 0601 08 23 05 30 01 12 00 00 00   CMD Jog_CCW
0005249371 0581 08 60 05 30 01 00 00 00 00        ACK
0005249376 0181 06 27 44 02 12 12 00              TxPDO1 z = 0, 18, 18, 2
0005249378 0181 06 27 40 00 12 12 00              TxPDO1 z = 0, 18, 18, 0
0005249379 0181 06 27 50 00 12 12 00              TxPDO1 z = 0, 18, 18, 0
0005249381 0181 06 27 54 00 12 12 00              TxPDO1 z = 0, 18, 18, 0
0005249470 0601 08 40 05 30 02 00 00 00 00   REQ
0005249471 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005249570 0601 08 40 05 30 02 00 00 00 00   REQ
0005249571 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005249670 0601 08 40 05 30 02 00 00 00 00   REQ
```

```
0005249671 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005249770 0601 08 40 05 30 02 00 00 00 00   REQ
0005249771 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005249870 0601 08 40 05 30 02 00 00 00 00   REQ
0005249871 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005249970 0601 08 40 05 30 02 00 00 00 00   REQ
0005249971 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005250070 0601 08 40 05 30 02 00 00 00 00   REQ
0005250071 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005250170 0601 08 40 05 30 02 00 00 00 00   REQ
0005250171 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18
0005250270 0601 08 40 05 30 02 00 00 00 00   REQ
0005250271 0581 08 43 05 30 02 12 00 00 00        (b) Funct Status = 18

0005250291 0601 08 23 05 30 01 00 00 00 00   CMD Cancel()
0005250292 0581 08 60 05 30 01 00 00 00 00        ACK
0005250293 0181 06 27 40 00 12 12 00              TxPDO1 z = 0, 18, 18, 0
0005250298 0181 06 27 04 00 00 12 01              TxPDO1 z = 1, 18, 0, 0
0005250370 0601 08 40 05 30 02 00 00 00 00   REQ
0005250371 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0005250371 0601 08 40 05 30 03 00 00 00 00   REQ
0005250373 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Function #19, Factory_Reset()

The sensor is returned to factory configuration

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000092240 0601 08 23 05 30 01 13 00 00 00   CMD Factory_Reset()
0000092241 0581 08 60 05 30 01 00 00 00 00        ACK
0000092251 0181 06 27 44 00 13 13 00              TxPDO1 z = 0, 19, 19, 0
No READY bit is sent
0000092340 0601 08 40 05 30 02 00 00 00 00   REQ
0000092341 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000092341 0601 08 40 05 30 03 00 00 00 00   REQ
0000092343 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000092350 0000 02 02 01                     NMT Stop
0000092360 0000 02 81 01                     NMT Reset Node
0000092613 0701 01 00                             BOOT (Node Id = 1)
0000001020 0601 08 40 02 30 00 00 00 00 00   REQ
0000001021 0581 08 4F 02 30 00 01 00 00 00      Autostart = 1
0000003020 0000 02 01 01                     NMT Operational
0000003021 0181 06 27 04 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
READY bit is sent
```

Function #20, ER_Parameters_Reset()

The sensor detection parameters are returned to factory settings.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000112740 0601 08 23 05 30 01 14 00 00 00   CMD ER_Parameters_Reset()
0000112741 0581 08 60 05 30 01 00 00 00 00        ACK
0000112750 0181 06 27 44 00 14 14 00              TxPDO1 z = 0, 20, 20, 0
0000112840 0601 08 40 05 30 02 00 00 00 00   REQ
0000112841 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000112841 0601 08 40 05 30 03 00 00 00 00   REQ
0000112842 0181 06 27 04 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0000112843 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Change Node ID

The device node ID can be changed through the following command execution sequence. In this example, the system is reset after the node ID change. This isn't required.

- Write new Node ID to OD location 0x2400.03

- Write SAVE signature to OD location 0x1010.01 (SAVE signature is ascii text “save”). Note that the ACK response from this write is about 200ms. This timing must be observed.
- Send the NMT Reset
- Observe the 3 second initialization time after Boot message is received. Same as a power up sequence.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000040930 0606 08 2F 00 24 03 04 00 00 00   CMD Generic Write (0x2400.03 = 4)
0000040931 0586 08 60 00 24 03 00 00 00 00        ACK
0000040940 0606 08 23 10 10 01 73 61 76 65   CMD SAVE Sig
0000042705 0586 08 60 10 10 01 00 00 00 00        ACK
0000045920 0000 02 81 00                     NMT Reset Node
0000046173 0704 01 00                             BOOT (Node Id = 4)
0000046180 0604 08 23 05 30 01 19 00 00 00   CMD Factory_Reset()
0000046181 0584 08 60 05 30 01 00 00 00 00        ACK
0000046280 0604 08 40 05 30 02 00 00 00 00   REQ
0000046281 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000046281 0604 08 40 05 30 03 00 00 00 00   REQ
0000046283 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000000272 0704 01 00                             BOOT (Node Id = 4)
Wait for 3 seconds…
0000003290 0000 02 01 04                     NMT Operational
0000003291 0184 06 27 00 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0000003300 0604 08 40 05 30 17 00 00 00 00   REQ
0000003301 0584 08 43 05 30 17 13 00 00 00        (w) Script Rev = 0.19
```

Scan for sensor parameters

A scan of operational parameters is shown. This is a read of OD 0x3005.00 to 0x3005.1A.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000349110 0601 08 40 05 30 00 00 00 00 00   REQ
0000349111 0581 08 4F 05 30 00 1A 00 00 00        (0) Constant = 26
0000349120 0601 08 40 05 30 01 00 00 00 00   REQ
0000349121 0581 08 43 05 30 01 00 00 00 00        (a) Function = 0
0000349130 0601 08 40 05 30 02 00 00 00 00   REQ
0000349131 0581 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000349140 0601 08 40 05 30 03 00 00 00 00   REQ
0000349141 0581 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000349150 0601 08 40 05 30 04 00 00 00 00   REQ
0000349151 0581 08 43 05 30 04 00 00 00 00        (d) Encoder Offset = 0
0000349160 0601 08 40 05 30 05 00 00 00 00   REQ
0000349161 0581 08 43 05 30 05 00 00 00 00        (e) Job Cycles = 0
0000349170 0601 08 40 05 30 06 00 00 00 00   REQ
0000349171 0581 08 43 05 30 06 67 03 00 00        (f) Life Cycles = 871
0000349180 0601 08 40 05 30 07 00 00 00 00   REQ
0000349181 0581 08 43 05 30 07 00 00 00 00        (g) Job Time = 0
0000349190 0601 08 40 05 30 08 00 00 00 00   REQ
0000349191 0581 08 43 05 30 08 1E 01 00 00        (h) Life Time = 286
0000349200 0601 08 40 05 30 09 00 00 00 00   REQ
0000349201 0581 08 43 05 30 09 00 00 00 00        (i) Reserved
0000349210 0601 08 40 05 30 0A 00 00 00 00   REQ
0000349211 0581 08 43 05 30 0A 00 00 00 00        (j) Reserved
0000349220 0601 08 40 05 30 0B 00 00 00 00   REQ
0000349221 0581 08 43 05 30 0B 00 00 00 00        (k) CW Target Pos = 0
0000349230 0601 08 40 05 30 0C 00 00 00 00   REQ
0000349231 0581 08 43 05 30 0C 32 00 00 00        (l) CW Velocity = 50
0000349240 0601 08 40 05 30 0D 00 00 00 00   REQ
0000349241 0581 08 43 05 30 0D C8 00 00 00        (m) CW Force = 200
0000349250 0601 08 40 05 30 0E 00 00 00 00   REQ
0000349251 0581 08 43 05 30 0E 1D 00 00 00        (n) CW Tolerance = 29
0000349260 0601 08 40 05 30 0F 00 00 00 00   REQ
0000349261 0581 08 43 05 30 0F 00 00 00 00        (o) CCW Target Pos = 0
0000349270 0601 08 40 05 30 10 00 00 00 00   REQ
0000349271 0581 08 43 05 30 10 32 00 00 00        (p) CCW Velocity = 50
```

```
0000349280 0601 08 40 05 30 11 00 00 00 00   REQ
0000349281 0581 08 43 05 30 11 C8 00 00 00        (q) CCW Force = 200
0000349290 0601 08 40 05 30 12 00 00 00 00   REQ
0000349291 0581 08 43 05 30 12 1D 00 00 00        (r) CCW Tolerance = 29
0000349300 0601 08 40 05 30 13 00 00 00 00   REQ
0000349301 0581 08 43 05 30 13 00 00 00 00        (s) Reserved
0000349310 0601 08 40 05 30 14 00 00 00 00   REQ
0000349311 0581 08 43 05 30 14 00 00 00 00        (t) Watchdog Flag = 0
0000349320 0601 08 40 05 30 15 00 00 00 00   REQ
0000349321 0581 08 43 05 30 15 00 00 00 00        (u) Reserved
0000349330 0601 08 40 05 30 16 00 00 00 00   REQ
0000349331 0581 08 43 05 30 16 00 00 00 00        (v) Cycle Timer Flag = 0
0000349340 0601 08 40 05 30 17 00 00 00 00   REQ
0000349341 0581 08 43 05 30 17 16 00 00 00        (w) Script Rev = 0.22
0000349350 0601 08 40 05 30 18 00 00 00 00   REQ
0000349351 0581 08 43 05 30 18 00 00 00 00        (x) Detect Delta = 0
0000349360 0601 08 40 05 30 19 00 00 00 00   REQ
0000349361 0581 08 43 05 30 19 01 00 00 00        (y) Status  = 0x00000001
0000349370 0601 08 40 05 30 1A 00 00 00 00   REQ
0000349371 0581 08 43 05 30 1A 00 00 00 01        (z) Monitor = 0x01000000
```

Set system variables

New sweep and tolerance parameters are set for the CW direction. These can optionally be saved to non-volatile memory

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0000519950 0604 08 23 05 30 0C 64 00 00 00   SET CW Velocity = 100
0000519951 0584 08 60 05 30 0C 00 00 00 00        ACK
0000519960 0604 08 23 05 30 0D 5E 01 00 00   SET CW Torque = 350
0000519961 0584 08 60 05 30 0D 00 00 00 00        ACK
0000519970 0604 08 23 05 30 0E 58 00 00 00   SET CW Tolerance = 88
0000519971 0584 08 60 05 30 0E 00 00 00 00        ACK
0000519980 0604 08 23 05 30 01 0F 00 00 00   CMD Detect_CW()
0000519981 0584 08 60 05 30 01 00 00 00 00        ACK
```

Setting Home and Target Positions Manually

The home and target positions for the sensor can be set and re-set manually. The use of the five functions to accomplish this are shown.

Execute “Hold_Zero_Speed(). This causes the sensor to switch modes and allow the wand to be repositioned

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003160310 0604 08 23 05 30 01 01 00 00 00   CMD Hold_Zero_Speed()
0003160311 0584 08 60 05 30 01 00 00 00 00        ACK
0003160318 0184 06 27 44 00 01 01 00              TxPDO1 z = 0, 1, 1, 0
0003160319 0184 06 27 40 00 01 01 00              TxPDO1 z = 0, 1, 1, 0
0003160320 0184 06 27 50 00 01 01 00              TxPDO1 z = 0, 1, 1, 0
0003160322 0184 06 27 54 00 01 01 00              TxPDO1 z = 0, 1, 1, 0
0003160325 0184 06 27 14 00 00 01 00              TxPDO1 z = 0, 1, 0, 0
0003160331 0184 06 27 54 00 00 01 01              TxPDO1 z = 1, 1, 0, 0
0003160410 0604 08 40 05 30 02 00 00 00 00   REQ
0003160411 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003160411 0604 08 40 05 30 03 00 00 00 00   REQ
0003160413 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0003160828 0184 06 27 14 00 00 01 01              TxPDO1 z = 1, 1, 0, 0
```

Manually rotate the wand to the new home position. Execute the Set_Home_Position(). There is no motion associated with this. This position is captured as the new home.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003277520 0604 08 23 05 30 01 05 00 00 00   CMD Set_Home()
0003277521 0584 08 60 05 30 01 00 00 00 00        ACK
0003277526 0184 06 27 00 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0003277527 0184 06 27 04 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0003277528 0184 06 27 44 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0003277529 0184 06 27 54 00 05 05 00              TxPDO1 z = 0, 5, 5, 0
0003277532 0184 06 27 44 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0003277533 0184 06 27 40 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0003277534 0184 06 27 50 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0003277535 0184 06 27 54 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0003277547 0184 06 27 14 00 00 05 00              TxPDO1 z = 0, 5, 0, 0
0003277552 0184 06 27 54 00 00 05 01              TxPDO1 z = 1, 5, 0, 0
0003277620 0604 08 40 05 30 02 00 00 00 00   REQ
0003277621 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003277621 0604 08 40 05 30 03 00 00 00 00   REQ
0003277623 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0003278052 0184 06 27 14 00 00 05 01              TxPDO1 z = 1, 5, 0, 0
```

Manually rotate the wand to the new clockwise target position. Execute the Teach_Manual_CW() function. There is no motion; the position is captured at the new target.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003455280 0604 08 23 05 30 01 15 00 00 00   CMD Teach_Manual_CW()
0003455281 0584 08 60 05 30 01 00 00 00 00        ACK
0003455285 0184 06 27 54 00 00 05 01              TxPDO1 z = 1, 5, 0, 0
0003455291 0184 06 27 14 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0003455380 0604 08 40 05 30 02 00 00 00 00   REQ
0003455381 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003455381 0604 08 40 05 30 03 00 00 00 00   REQ
0003455383 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

Rotate the wand to the new counter-clockwise target position. Execute the Teach_Manual_CCW() function. There is no motion; the position is captured at the new target.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003529620 0604 08 23 05 30 01 16 00 00 00   CMD Teach_Manual_CCW
0003529621 0584 08 60 05 30 01 00 00 00 00        ACK
0003529624 0184 06 27 54 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0003529629 0184 06 27 14 00 00 00 01              TxPDO1 z = 1, 0, 0, 0
0003529720 0604 08 40 05 30 02 00 00 00 00   REQ
0003529721 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003529721 0604 08 40 05 30 03 00 00 00 00   REQ
0003529723 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
```

The Go_Home() function is executed. The sensor powers up and moves to the new home position. It is now ready to execute detect functions to the new targets.

```
Time (ms)  COB DLC DATA                          DESCRIPTION
0003565570 0604 08 23 05 30 01 06 00 00 00   CMD Go_Home()
0003565571 0584 08 60 05 30 01 00 00 00 00        ACK
0003565576 0184 06 27 00 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003565577 0184 06 27 40 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003565579 0184 06 27 50 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003565580 0184 06 27 40 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003565670 0604 08 40 05 30 02 00 00 00 00   REQ
0003565671 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0003565770 0604 08 40 05 30 02 00 00 00 00   REQ
0003565771 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0003565870 0604 08 40 05 30 02 00 00 00 00   REQ
0003565871 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0003565970 0604 08 40 05 30 02 00 00 00 00   REQ
0003565971 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
```

```
0003566038 0184 06 27 44 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003566043 0184 06 27 04 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0003566049 0184 06 27 44 00 00 06 01              TxPDO1 z = 1, 6, 0, 0
0003566070 0604 08 40 05 30 02 00 00 00 00   REQ
0003566071 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0003566071 0604 08 40 05 30 03 00 00 00 00   REQ
0003566073 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0003566541 0184 06 27 04 00 00 06 01              TxPDO1 z = 1, 6, 0, 0
```

System Faults

System faults result from the failure of a function to execute properly. An example of this is the case where a detect cycle is started but fails to return to its home position. After a timeout period of 10 seconds, a failure is reported by a 255 in the Function Status Register (0x3005.02) and by the TxPDO1 Ready bit assertion with a 255 fault code.

At this point, the sensor is in an un-powered state to prevent overheating.

```
Time (ms)  COB DLC DATA                         DESCRIPTION
0000016830 0604 08 23 05 30 01 0F 00 00 00   CMD Detect_CW()
0000016831 0584 08 60 05 30 01 00 00 00 00        ACK
0000016840 0184 06 27 44 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000016842 0184 06 27 50 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000016843 0184 06 27 40 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000016930 0604 08 40 05 30 02 00 00 00 00   REQ
0000016931 0584 08 43 05 30 02 0F 00 00 00        (b) Funct Status = 15
0000017030 0604 08 40 05 30 02 00 00 00 00   REQ
0000017031 0584 08 43 05 30 02 0F 00 00 00        (b) Funct Status = 15
0000017033 0184 06 27 50 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000017034 0184 06 27 54 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000017035 0184 06 27 40 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000017038 0184 06 27 50 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000017039 0184 06 27 40 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000017130 0604 08 40 05 30 02 00 00 00 00   REQ
0000017131 0584 08 43 05 30 02 0F 00 00 00        (b) Funct Status = 15
Continued polling ---

0000026930 0604 08 40 05 30 02 00 00 00 00   REQ
0000026931 0584 08 43 05 30 02 0F 00 00 00        (b) Funct Status = 15
0000027030 0604 08 40 05 30 02 00 00 00 00   REQ
0000027031 0584 08 43 05 30 02 0F 00 00 00        (b) Funct Status = 15
0000027040 0184 06 27 50 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000027042 0184 06 27 54 00 0F 0F 00              TxPDO1 z = 0, 15, 15, 0
0000027045 0184 06 27 14 00 FF 0F 00              TxPDO1 z = 0, 15, 255, 0
0000027050 0184 06 27 54 00 FF 0F 01              TxPDO1 z = 1, 15, 255, 0
0000027130 0604 08 40 05 30 02 00 00 00 00   REQ
0000027131 0584 08 43 05 30 02 FF 00 00 00        (b) Funct Status = 255 - FAIL
0000027140 0604 08 40 05 30 03 00 00 00 00   REQ
0000027141 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000027540 0184 06 27 14 00 FF 0F 01              TxPDO1 z = 1, 15, 255, 0
The READY bit is asserted here. This indicates that the unit is ready to accept the commands needed to
recover from the fault
```

System Fault Recovery

After a system fault has been asserted, the recovery process requires that the function in progress is cancelled and the result register cleared. Then a new function can execute. In this case a Go_Home() is executed and the function terminates properly.

```
Time (ms)  COB DLC DATA                         DESCRIPTION
```

```
0000046629 0604 08 23 05 30 01 00 00 00 00   CMD Cancel()
0000046630 0584 08 60 05 30 01 00 00 00 00        ACK
0000046630 0604 08 23 05 30 02 00 00 00 00   CMD Clear Function Status
0000046632 0584 08 60 05 30 02 00 00 00 00        ACK
Ready for a new function----

0000046640 0604 08 23 05 30 01 06 00 00 00   CMD Go_Home()
0000046641 0584 08 60 05 30 01 00 00 00 00        ACK
0000046645 0184 06 27 00 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000046646 0184 06 27 40 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000046648 0184 06 27 50 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000046649 0184 06 27 40 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000046740 0604 08 40 05 30 02 00 00 00 00   REQ
0000046741 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0000046840 0604 08 40 05 30 02 00 00 00 00   REQ
0000046841 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0000046940 0604 08 40 05 30 02 00 00 00 00   REQ
0000046941 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0000047040 0604 08 40 05 30 02 00 00 00 00   REQ
0000047041 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0000047140 0604 08 40 05 30 02 00 00 00 00   REQ
0000047141 0584 08 43 05 30 02 06 00 00 00        (b) Funct Status = 6
0000047151 0184 06 27 44 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000047156 0184 06 27 04 00 06 06 00              TxPDO1 z = 0, 6, 6, 0
0000047162 0184 06 27 44 00 00 06 01              TxPDO1 z = 1, 6, 0, 0
0000047240 0604 08 40 05 30 02 00 00 00 00   REQ
0000047241 0584 08 43 05 30 02 00 00 00 00        (b) Funct Status = 0 - OK
0000047241 0604 08 40 05 30 03 00 00 00 00   REQ
0000047243 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000047250 0604 08 40 05 30 03 00 00 00 00   REQ
0000047251 0584 08 43 05 30 03 00 00 00 00        (c) Funct Result = 0 - OK
0000047660 0184 06 27 04 00 00 06 01              TxPDO1 z = 1, 6, 0, 0
```

Free Space Monitoring

Normal detection functions can be used to implement free space validation. A target position is established at a position beyond the expected obstruction by executing normal teach functions. The teach sweep will travel until a target is detected. Subsequent detect cycles will normally travel to the target and report a ‘0’ (OK), or report a ‘1’ (Under-travel) condition.

Since this is process doesn’t normally involve a delicate target or obstruction, high force and velocity values can be used to speed up the process.

Communication Bus Efficiency

Bus traffic and communication overhead can be reduced by implementing function execution via RxPDO_1 instead of SDO writes to 0x3005.01. By default, the sensors are configured with RxPDO_1 = (COBID 0x200+Node_Id).

By default, the transmission type is set to 255 so execution will begin immediately on reception. RxPDO_1 is mapped for a single 32bit variable. This can be altered by normal CANopen procedures. The transmission type can be configured to ‘0’ (synchronous type) so that multiple units can be loaded and started simultaneously with a SYNC signal.

Also, to reduce the number of TxPDO1 messages, the transmission type for that can be set to synchronous. The status will then be sent with a SYNC signal.

At a 500kbps communications rate, it takes about 0.52ms to send the command.

Example of sending Function#1 (Hold_Zero_Speed()) then sending Function #6 (Go_Home()) to node ID 4 using RxPDO_1 mapping

```
Time (ms)  COB DLC DATA                        DESCRIPTION
0003642340 0204 04 01 00 00 00              RXPDO1 Function = 1 (Hold_Zero_Speed())
0003642349 0184 06 27 44 00 01 01 00             TxPDO1 z = 0, 1, 1, 0
0003642350 0184 06 27 40 00 01 01 00             TxPDO1 z = 0, 1, 1, 0
0003642351 0184 06 27 50 00 01 01 00             TxPDO1 z = 0, 1, 1, 0
0003642352 0184 06 27 54 00 01 01 00             TxPDO1 z = 0, 1, 1, 0
0003642356 0184 06 27 14 00 00 01 00             TxPDO1 z = 0, 1, 0, 0
0003642362 0184 06 27 54 00 00 01 01             TxPDO1 z = 1, 1, 0, 0
0003642854 0184 06 27 14 00 00 01 01             TxPDO1 z = 1, 1, 0, 0

0003730220 0204 04 06 00 00 00              RXPDO1 Function = 6 (Go_Home())
0003730226 0184 06 27 00 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730227 0184 06 27 40 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730229 0184 06 27 50 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730230 0184 06 27 40 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730234 0184 06 27 44 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730239 0184 06 27 04 00 06 06 00             TxPDO1 z = 0, 6, 6, 0
0003730245 0184 06 27 44 00 00 06 01             TxPDO1 z = 1, 6, 0, 0
0003730737 0184 06 27 04 00 00 06 01             TxPDO1 z = 1, 6, 0, 0
```

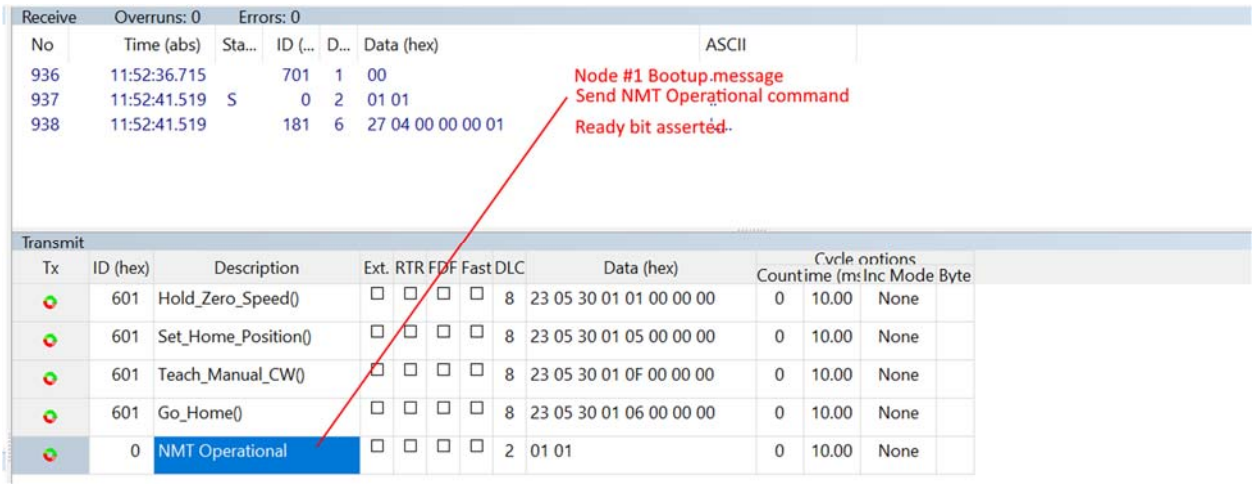
Quick Start Tutorial

These examples assume there is only one sensor device on the network and its Node ID is known. If the Node ID is unknown, apply power the unit. At power up, it will transmit the CANopen 'Boot' message which carries the Node ID.

Functions from the command library can be executed manually. These can be performed using most any CAN bus tool. The examples below are done with the "canAnalyzer3 mini" software tool from IXXAT. It is included with the purchase of their PC-based CAN interface hardware.

Power Up

At power up the sensor transmits the CanOpen "Bootup" message. The host sends NMT Operational command.



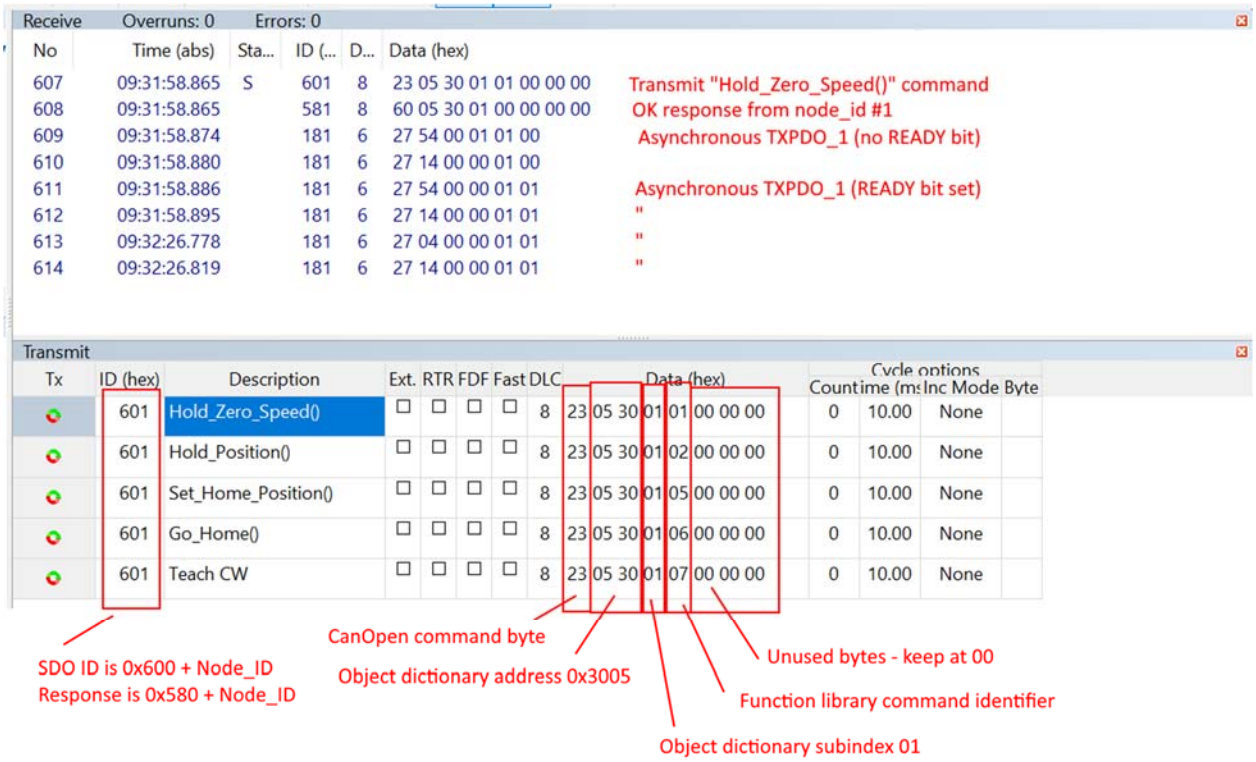
Example A – Home position management

A configuration to execute four library functions is shown. This series of instructions can be used to manually establish a new home position then return to it.

Apply power to the unit. It will remain in its position without moving. If the wand is moved manually, it will return.

Execute ‘Hold_Zero_Speed()’. The wand can be moved manually and it will remain in position. Rotate the wand to a new position which will be the new ‘home’. Execute the ‘Set_Home_Position()’ command. This position is established as the new home; the wand can still be moved without resistance. Move the wand to a new position away from the new home. Execute the Go_Home() command. The wand will return to the established home position and remain there.

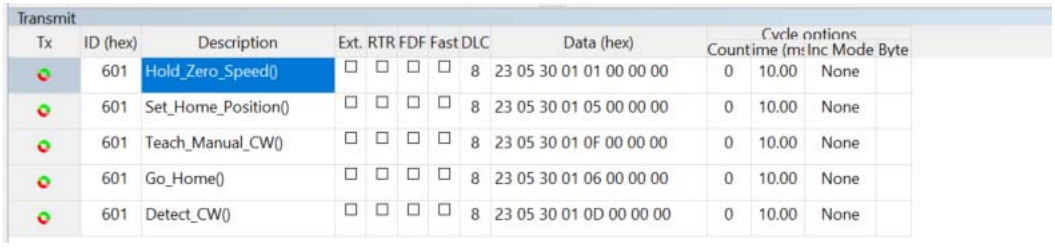
The other function in this group will power the wand and hold it in a position, but does not establish the position as ‘home’.



Example B – Manual teach home and target positions

This example executes the functions necessary to establish operating positions by manually rotating the wand into the desired locations. These locations can then be used as target positions for subsequent detect cycles.

Note that the Clockwise/Counterclockwise orientation is viewed from the wand end of the sensor



The Hold_Zero_Speed() function library command is executed. This will release the drive control and allow the wand to be moved without returning to its previous position.

Rotate the wand to the position which will be designated as the ‘home’ position. Execute the Set_Home_Position() function. This will only establish the position. The wand can continue to be moved.

Rotate the wand to the CW target position. Execute the Teach_Manual_CW.

Execute the Go_Home() function. The wand will move to the established position and hold there.

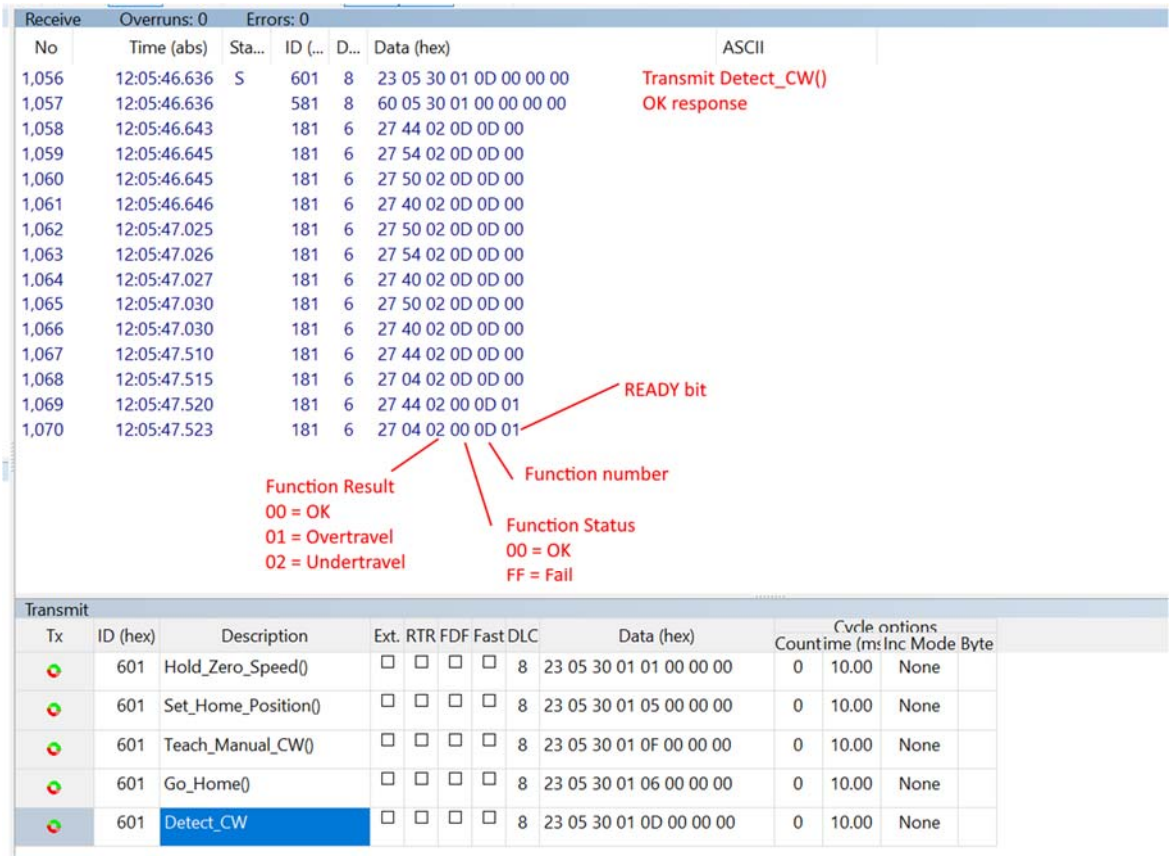
These positions are automatically saved in non-volatile memory and will be retained until new positions are established (including power-cycle).

Example C – Testing target positions

From Example B, the target positions can now be used with the detect functions.

The Detect_CW() function is executed. While the function is in progress, the sensor sends asynchronous messages until the function completes. The READY bit is set and the function result is presented.

The Detect_CCW function execution is the same except that it will return 0x0E while in progress.



Appendix A: Asynchronous System and Function Status Communications

Sensor system firmware manipulates bit #14 in the statusword OD entry (0x6041) to cause the 0x3005.1A 'Tool Monitor' variable to be transmitted asynchronously via TxPDO1 (Event Triggered).

After power-up (when ready to accept commands), the Ready Flag of the Tool Monitor word will be set. All other bytes are cleared. Then Bit#14 of statusword will be set. TxPDO1 will transmit 0x01000000.

Function execution starts with a SDO write to 3005.01 (Teach_CW for example). The Ready Flag (bit24) is cleared, the function number (0x07 in this case) is written to Byte_1 and Byte_2, Byte_0 is cleared. Statusword bit #14 is cleared, triggering TxPDO1 to transmit 0x00070700. Because the Ready Flag is cleared, no additional function may be initiated.

At completion of function execution (some will take many milliseconds to complete):

- The execution result code is written into Byte_0
- Byte_1 is set to zero (assuming execution completed properly)
- Byte_2 remains at 0x07
- The Ready Flag is set. Statusword bit#14 is set, again triggering TxPDO1
- Statusword bit#14 is cleared in preparation for the next command. This transition also causes a second (duplicate) TxPDO1 to be transmitted
- The sensor is now ready to receive another command.

Function status and result codes are defined elsewhere in this document.

Byte_3 (MSB)								Byte_2								Byte_1								Byte_0 (LSB)							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused							0	Function in progress								Execution status								Execution result code							

System status bit
1 = Ready Flag
0 = Not Ready

Appendix B: Cable Shielding

To be effective in their role as a noise reduction mechanism, shields need proper termination to earth ground.

https://www.icotek.com/en-us/products/emc-cable-shield-clamps/sf

Revision Info

Rev	Date	By
1.00	Initial release	