Orca™ Series Modbus

User Guide 210912

Version 1.2

This document applies to the following Orca Series motor firmware:

- 6.1.4
- 6.1.5
- 6.1.6
- 6.1.7

For more recent firmware versions, please download the latest version of this user guide at https://irisdynamics.com/downloads

CONTENTS

Revision History

Version	Date	Author	Reason
1.0	September, 2021	kh ke rm	Initial Release
1.1	March, 2023	Kh kc rm	Memory map, frame examples, formatting
1.2	August, 2023	rm	Kinematic mode, frame examples, sub function codes, additional stream commands. CRC calculation

Introduction

Orca Series Motors feature a 'field-bus' serial communication option which allows configuration, control, and monitoring. Features of the motors are offered by exposing 'registers' which can be written to and read from by sending and receiving characters.

The motor only responds to communications: it will never send characters except when responding to a well-received message.

In this documentation, 'Tx' (*i.e.,* transmitter / transmitting) refers to data sent from the motor to the controller. A controller in this case is a PLC or some computer which is responsible for controlling the motor. 'Rx' (*i.e.,* receiver / receiving) refers to data sent from the controller to the motor.

Serial communications are implemented using a subset of the Modbus RTU specification, with additional functionality to support a high-speed stream of commands and feedback.

Physical Interface

The signaling used for transmitting characters conforms to and exceeds the RS422 specification.

Orca Series motors include a shielded communication cable of twisted pairs carrying the differential signals used to transmit and receive characters. The connection is full duplex and so there is a separate pair for Tx and Rx. For half-duplex communication see the "Orca Series Modbus over Half-Duplex RS485" user guide.

Each motor data cable will have the following signals:

Table 1: Orca Series Motor RJ45 Pinout

Pin	Signal	Notes
1	RX_2+	Modbus
2	RX_2-	
3	TX_2+	
4	RX/TX_1+	IrisControls4™
5	RX/TX_1-	
6	TX_2-	Modbus
7	+5V	Can be used to power logic, or indicate logic power status.
8	GND	Must be connected to the communication hub ground.

Modbus RTU Protocol

The motor implements a Modbus RTU server with the following initial requirements:

Baud Rate: 19200
Start Bits: 1
Stop Bits: 1
Parity: even
Interframe delay: 2 ms

For information on the specifics of Modbus message framing, see MODBUS_ Application Protocol 1.1b.

The motors also allow for configuration of the initial baud rate and interframe delay.

Orca Series Motors implement the following MODBUS RTU function codes.

Table 2: Function Codes Supported by Orca Series Motors

Function Code Type	Name	Code (Decimal / Hex)
Standard Function Codes	Read Holding Registers	3 / 0x03
	Write Single Register	6 / 0x06
	Write Multiple Registers	16 / 0x10
	Diagnostic: Return Query Data	8 / 0x08 Sub code: 0 / 0x0000
Orca-Specific Function Codes	Manage High-speed Stream	65 / 0x41
	Motor Command Stream	100 / 0x64
	Motor Read Stream	104 / 0x68
	Motor Write Stream	105 / 0x69

Most registers are 16-bit. Some registers are 32-bit, these are always encoded with the most significant bits in the higher register address (*i.e.,* little-endian).

Modbus commands are used to access the registers within the motor. Any configurations that can be made through the Orca Series motor's GUI can also be made through standard Modbus write commands. A full list of all Orca Series motor registers is available in "Orca API User Guide".

Orca Series motors use 0 indexing, meaning the first register address is 0. Some Modbus libraries however use 1 indexing, meaning the first register's address is 1. If this is the case the address in the Orca's Memory map will need to be adjusted accordingly. For example if your Modbus client uses 1 indexing and you would like to change the motor's mode, write to register 4 rather than 3 for CTRL_REG_3.

Orca-specific Function Codes

Modbus is a connection-less specification and as such, messages can be sent to the motor at irregular intervals. However, motor functions that result in forces being produced (*i.e* . Force control and Position control) require that a consistent stream of well-formed messages are received regularly, or else the motor enters a Sleep Mode. The value in the Timeout Control Register determines how long a Force or Position command remains valid.

To accommodate streaming low-latency messages at higher data rates, Orca-specific function codes allow establishing a connection during which the baud rate can be increased, and interframe delays can be decreased beyond Modbus specifications.

Use of these is optional but will result in higher data throughput and lower data latency on the serial connection.

65 / 0x41 Manage High-speed Stream

This function code is used to enable or disable a high-speed stream, and when enabling it, to specify the parameters of subsequent messages.

Table 3: Manage High-speed Stream Request PDU

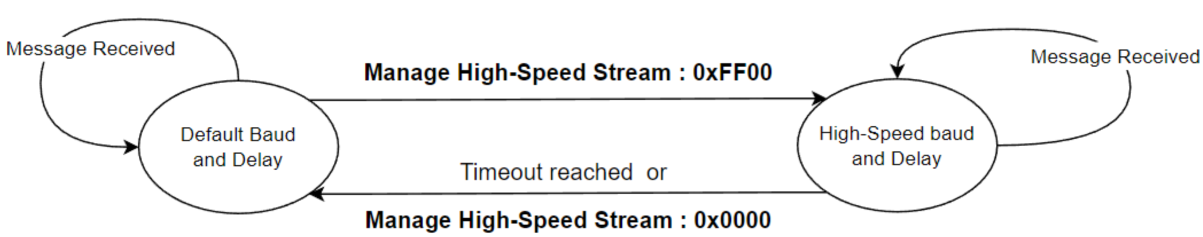
Device Address	1 byte	0x01	
Function Code	1 byte	0x41	
Sub-Function Code	2 bytes	enable and apply parameters 0xFF00	disable and return to default 0x0000
Baud rate	4 bytes	Target baud rate (bps)	Ignored
Delay (us)	2 bytes	Target Response delay (ms)	Ignored
CRC	2 bytes	CRC-16 (MODBUS) Polynomial 0xA001	

Table 4: Manage High-speed Stream Response PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x41
State Command	1 byte	Echo of request
Baud rate	4 bytes	Realized baud rate
Delay (us)	2 bytes	Realized message delay in microseconds
CRC	2 bytes	CRC-16 (MODBUS) Polynomial 0xA001

The baud rate and response delay will return to their default setting when either:

1. Time elapses greater than the value in the Timeout Control Register after the last successful message
2. A new message is sent to disable the high-speed stream



100 / 0x64 Motor Command Stream

This function code is used to stream commands and a specified operating mode to the motor while receiving several of the important motor sensor information and errors in return. It is expected that this function code is streamed at as high a frequency as possible to ensure the client has recent information on the motor, and the motor has a recent command to act on.

The data that must be a part of the Stream Command request varies between sub-function codes. However, the response frame remains the same for all sub-function codes.

Table 5: Stream Command Request PDU

Device Address	1 byte	0x01				
Function Code	1 byte	0x64				
Sub Code	1 byte	0x1C Force Control Stream	0x1E Position Control Stream	0x20 Kinematic Data Stream *(Available with Orca firmware v6.1.6 or later)*	0x22 Haptic Data Stream *(Available with Orca firmware v6.1.7 or later)*	All else Sleep Data Stream
Data	4 bytes	Force (mN)	Position (µm)	Ignored	HAPTIC_STATUS Register	Ignored
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001				

Table 6: Stream Command Response PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x64
Position Value (µm)	4 bytes	Shaft position in micrometers
Force Value (mN)	4 bytes	Force realized in millinewtons
Power Value (W)	2 bytes	Power consumed in Watts
Temperature Value (C)	1 byte	Temperature value in degrees Celsius
Voltage Value (mV)	2 bytes	Supply Voltage in millivolts
Errors	2 bytes	Error register contents
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001

The motor will continue to act on the last Stream Command until

1. A new *100 / 0x64 Stream Command* message is received
2. One of the following registers are written to:
   a. the Position Control Register, or,
   b. the Force Control Register, or
   c. the Mode Control Register
3. The number of milliseconds equal to the value of the Timeout Control Register elapses from the last *100 / 0x64 Stream Command*.

Stream Command Sub-Function Codes

Orca Series motors can be operated in different modes which are described in detail in the Orca Series Motor Reference Manual.

The following sub-function codes are supported for the Stream Command which are related to the Orca Series motor's Modes of Operation.

1. Sleep Data Stream
2. Force Control Stream
3. Position Control Stream
4. Kinematic Data Stream *(Available with Orca firmware v6.1.6 or later)*
5. Haptic Data Stream *(Available with Orca firmware v6.1.7 or later)*

1 – Sleep Data Stream

Sending a Sleep Data Stream command will move the motor to Sleep Mode while allowing continuous stream of data from the motor to be maintained. In this mode the motor will only produce an electro-mechanical 'braking' force induced by shorting all its windings. No other force generation will be possible, regenerative braking is disabled, and motor power consumption will be minimized.

The motor will remain in "Sleep Mode" until a new mode is specified.

2 – Force Control Stream

Using the Force Control Stream sub-function code will put the motor into Force Mode. The force value in mN specified in the data portion of this command will be the target the motor attempts to reach.

A feedback loop adjusts power to the stator which continually accounts for temperature and voltage changes and shaft movement to achieve the target force.

If the time between Force Control Stream messages is greater than the value in the Timeout Control Register, the motor will enter Sleep Mode.

3 – Position Control Stream

Using the Position Control Stream sub-function code will put the motor into Position Mode. The position value in μm specified in the data portion of this command will be the target the motor will seek to achieve.

A tunable PID feedback loop adjusts forces used to attempt to position the shaft to match the target position.

The position control PID parameters can be configured by writing to (and optionally saving) configuration registers.

If the time between Position Control Stream messages is greater than the value in the Timeout Control Register, the motor will enter Sleep Mode.

4 – Kinematic Data Stream (Available with Orca firmware v6.1.6 or later)

The Kinematic Data Stream sub-function code is meant to enter the motor into Kinematic Mode while allowing continuous stream of data from the motor to be maintained. In Kinematic Mode the motor will use its position control to move between configured position targets.

These motions can be configured through the Orca Series motor's GUI in IrisControls4 or by writing to kinematic configuration registers (see the Orca Series Modbus Kinematics Triggering and Configuring User Guide). Triggering motions and configuring motions can be done by injecting additional Write Register Commands between Stream Commands.

The motor will remain in Kinematic Mode until a new mode is specified as the Timeout Control Registers do not apply to this mode.

5 – Haptic Data Stream *(Available with Orca firmware v6.1.7 or later)*
The Haptic Data Stream sub-function code is used to put the motor into Haptic Mode while allowing continuous stream of data from the motor to be maintained. In Haptic Mode the motor will use a set of configured Haptic effects to produce forces. This sub-function code includes a 2 byte data field which is used to enable and disable individual Haptic effects.

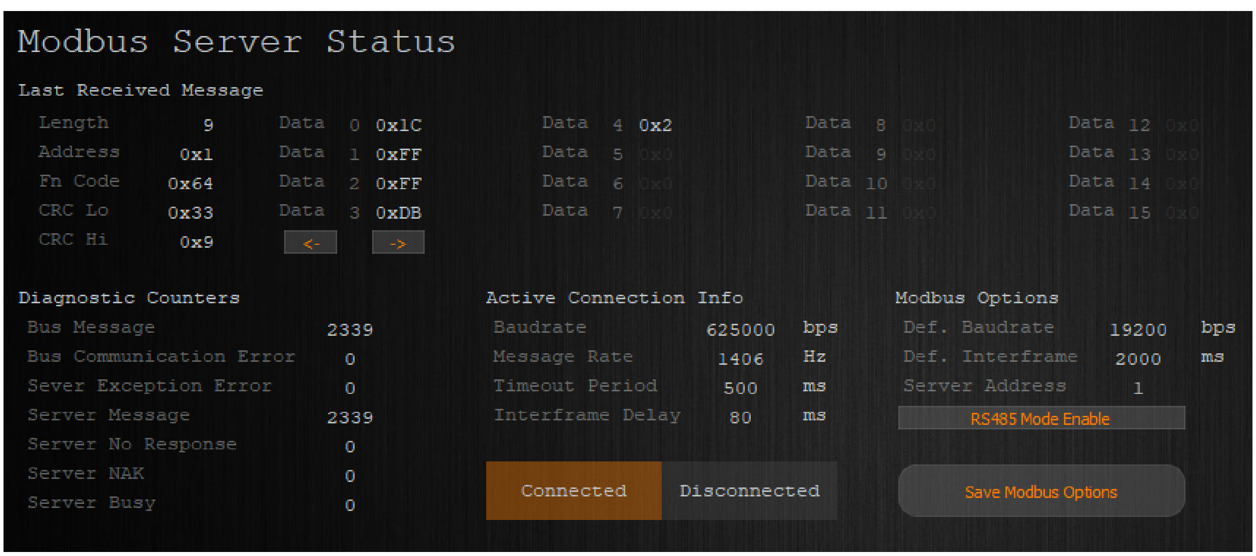
The Haptic effect parameters can be configured through the Orca Series motor's GUI in IrisControls4, or by writing to the Haptic configuration registers through additional Write Register Commands injected between Stream Commands.

Troubleshooting

Factors preventing the motor from achieving the target force or position include:

1. Commands exceeding the Max Force Control Register.
2. Power draw exceeding the Max Power Control Register.
3. Insufficient supply voltage.
4. Errors such as over-temperature or no shaft.
5. In Position Mode or Kinematic Mode appropriate PID tuning values must be set.
6. Force or Position Stream commands not sent fast enough resulting in a stream timeout.

The Orca Series motor's GUI has an Interfaces page available that gives data on the Modbus communication that can be useful for troubleshooting communication issues.



104 / 0x068 Motor Read Stream

This function code is used to stream a read from a register that is either single or double wide while receiving several of the important motor sensor information and errors in return as well as the current operating mode. Unlike the Motor Command Stream function code, this one will not explicitly set the mode. Setting the mode can be done by writing to Control Register 3, sending the motor read stream message will allow the motor to stay in the mode that was set. It is expected that this function code is streamed at a high frequency to ensure the client has recent information on the motor.

Table 7: Motor Read Stream Request PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x68
Register Address	2 bytes	The address of the register in the motor's memory map to read from
Register Width	1 byte	Specifies 1 if single wide register (16 bits of data) or use 2 if register is double wide (32 bits of data)
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001

Table 8: Motor Read Stream Response PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x68
Read Register Value	4 bytes	Value read from specified registers. If width was specified as 1 first 2 bytes will be zeros. If width of 2 then all bytes are used.
Mode of Operation	1 byte	Current operating mode of the Orca motor
Position Value (μm)	4 bytes	Shaft position in micrometers
Force Value (mN)	4 bytes	Force realized in millinewtons
Power Value (W)	2 bytes	Power consumed in Watts
Temperature Value (C)	1 byte	Temperature value in degrees Celsius
Voltage Value (mV)	2 bytes	Supply Voltage in millivolts
Errors	2 bytes	Error register contents
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001

105 / 0x069 Motor Write Stream

This function code is used to stream a write to a register that is either single or double wide while receiving several of the important motor sensor information and errors in return as well as the current operating mode. Unlike the Motor Command Stream function code, this one will not explicitly set the mode. Setting the mode can be done by writing to Control Register 3, sending the motor write stream message will allow the motor to stay in the mode that was set. It is expected that this function code is streamed at a high frequency to ensure the client has recent information on the motor. If a register is only needed to be written to once, the standard Modbus write single register or write multiple registers can be used.

Table 9: Motor Write Stream Request PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x69
Register Address	2 bytes	The address of the register in the motor's memory map to write to
Register Width	1 byte	Specifies 1 if single wide register (16 bits of data) or use 2 if register is double wide (32 bits of data)
Register Data	4 bytes	If width is specified as 1 the first 2 bytes will be ignored.
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001

Table 10: Motor Write Stream Response PDU

Device Address	1 byte	0x01
Function Code	1 byte	0x69
Mode of Operation	1 byte	Current operating mode of the Orca motor
Position Value (μm)	4 bytes	Shaft position in micrometers
Force Value (mN)	4 bytes	Force realized in millinewtons
Power Value (W)	2 bytes	Power consumed in Watts
Temperature Value (C)	1 byte	Temperature value in degrees Celsius
Voltage Value (mV)	2 bytes	Supply Voltage in millivolts
Errors	2 bytes	Error register contents
CRC	2 bytes	CRC-16 (Modbus) Polynomial 0xA001

Modbus Timeouts

Under certain modes of operation, Modbus messages sent to an Orca motor will reset a communication timer. If this timer is allowed to expire, i.e. no new Modbus messages arrive within the timeout period, a communications timeout error (2048) will be raised in the ERROR_0 and ERROR_1 registers.. This error prevents the motor from producing any forces (see RM220115 - Orca Series Reference Manual for more information on Orca errors). Note that the motor will remain in whatever mode it was last in, and the error can be cleared by returning to Sleep mode (1).

By default, the communication timeout period is set to 500 ms. The modes in which the communications timeout is active are:

- Force Mode (2)
- Position Mode (3)
- Haptic Mode (4)
- OAI Force Mode (8)
- OAI Position Mode (9)
- OAI Kinematic Mode (10)

Modes 8, 9 and 10 are used when connected to an Orca Analog Interface (OAI), not during regular Modbus communication modes (more information on the OAI can be found in UG230424 - Orca Analog Interface User Guide).

The communications timeout period can be shortened by writing a value between 0 and 500 ms to the USER_COMMS_TIMEOUT register. Writing values higher than 500 will result in a timeout period of 500 ms. To save this value permanently, the user options section of flash memory must be saved through CTRL_REG_2 (see RM220115 - Orca Series Reference Manual for more information).

Example Frames

Read Single Register Frame

Read one register 338 VDD Final. (8-byte message)
01 03 01 52 00 01 24 27
Two bytes give the content of the voltage register indicating 24267 mV being supplied. (7-byte message)
01 03 02 5E CB C1 B3

Write Single Register Frame

Write register 139 (User Max Temperature) to 60. (9-byte message)
01 06 00 8B 00 3C F9 F1
Response, echo indicates it is correctly written to. (9-byte message)
01 06 00 8B 00 3C F9 F1

Manage High-speed Stream Frame

Set high-speed stream baud rate to 625 kHz and 50 us interframe delay (12-byte message)
01 41 FF 00 00 09 89 68 00 32 A4 C1
Response, realized stream baud rate 625 kHz and 50 us interframe delay (12-byte message)
01 41 FF 00 00 09 89 68 00 32 A4 C1

Stream Command Frame

Sleep Data Stream request (9-byte message)
01 64 00 00 00 00 00 03 E4
Sleep Data Stream response (19-byte message)
01 64 00 03 89 65 00 00 06 BE 00 00 19 0F 01 00 00 88 C2

Force Control Stream request set force to 1000. (9-byte message)
01 64 1C 00 00 03 E8 D2 98
Force Control Stream Response, shaft position - 12000 µm, force - 800 , power - 20 W, temperature - 24 C, voltage - 24150 mV, errors - none (19-byte message)
01 64 00 01 3A 09 FF FB 52 00 00 18 5E B0 09 00 8F 44

Read Multiple Register Frame

Read two registers starting at address 406. (8-byte message)
01 03 01 96 00 02 25 D8
Response with content of two registers, 4 data bytes, Serial low is 53083, Serial high is 3373. This gives an overall serial number of 221106011. (9-byte message)
01 03 04 CF 5B 0D 2D 70 79

Write Multiple Register Frame

Write three registers starting at address 780. Setting kinematic motion 1 10000 µm and time of 1000ms. (15-byte message)
01 10 03 0C 00 03 06 27 10 00 00 03 E8 EE 51
Response with starting address and number of registers written to. (8-byte message)
01 10 03 0C 00 03 40 4F

CRC Calculation

The CRC bytes are used as the last two bytes in the message. They are used to ensure all bytes of the message have been communicated fully and accurately. All bytes of the message not including the two CRC bytes are used in calculating the CRC bytes.

Below is an example code snippet in C to generate the crc bytes.

C/C++

```
// Compute the MODBUS RTU CRC
UInt16 ModRTU_CRC(byte[] buf, int len)
{
  UInt16 crc = 0xFFFF;

  for (int pos = 0; pos < len; pos++) {
    crc ^= (UInt16)buf[pos];        // XOR byte into least sig. byte
of crc

    for (int i = 8; i != 0; i--) {// Loop over each bit
      if ((crc & 0x0001) != 0) {  // If the LSB is set
        crc >>= 1;                // Shift right and XOR 0xA001
        crc ^= 0xA001;
      }
      else                        // Else LSB is not set
        crc >>= 1;                // Just shift right
    }
  }
  // Note, this number has low and high bytes swapped, so use it
accordingly (or swap bytes)
  return crc;
}
```