KAWR

Contents

KAWR Driver

KAWR Driver for Kawasaki Robot (version 1.3, last revised 28 June 2018).

The KAWR driver enables communication between the Studio system and remote devices using the AS Language protocol, according to the specifications discussed in this document.

This document assumes that you have read the "Development Environment" section in the main Studio documentation.

This document also assumes that you are familiar with the Microsoft Windows XP/Vista/7 environment. If you are not familiar with Windows, then we suggest using the **Help and Support** feature (available from the Windows **Start** menu) as you work through this document.

Driver specifications

This section identifies all of the software and hardware components required to implement communication between the KAWR driver in Studio and remote devices using the AS Language protocol.

Driver files

The KAWR driver package comprises the following files, which are automatically installed in the `Drv` folder of the Studio application directory:

- `KAWR.DLL`: Compiled driver.
- `KAWR.INI`: Internal driver file. *You must not modify this file.*
- `KAWR.MSG`: Internal driver file defining error messages for the possible error codes. (These error codes are described in detail in the Troubleshooting section.) *You must not modify this file.*
- `KAWR.PDF`: This document, which provides complete information about using the driver.

> **Note:** You must use a compatible PDF reader to view the `KAWR.PDF` file. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's website.

You can use the KAWR driver on the following operating systems:

- Windows XP/Vista/7/8/2008/2012

Device specifications

To establish communication, your target device must meet the following specifications:

- Manufacturer: Kawasaki Robotics
- Compatible Equipment: Kawasaki D & E Controllers
- Programmer Software:

Network specifications

To establish communication, your device network must meet the following specifications:

- Device Communication Port: 23
- Physical Protocol: Telnet, TCP/IP
- Logic Protocol: AS Language
- Device Runtime Software: None
- Specific PC Board: None
- Cable Wiring Scheme: Regular Ethernet cable

Conformance testing

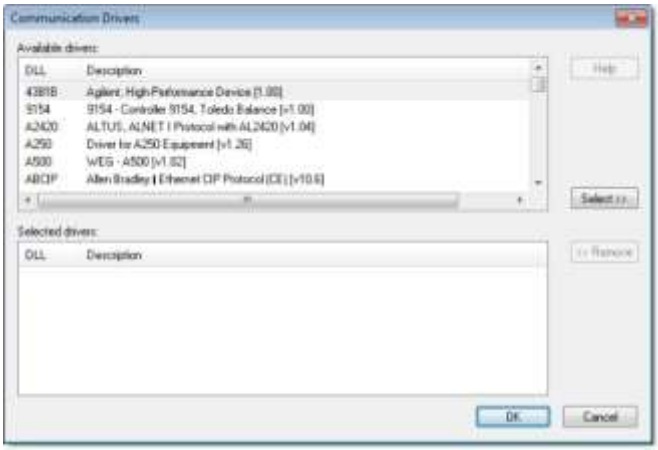
The following hardware/software was used for conformance testing:

- Driver Version: 1.3
- Studio Version: 8.1+SP1
- Operating System (development): Windows 7/8
- Operating System (target): Windows XP/Vista/7/8/2008/2012
- Equipment: Kawasaki D & E Controller Robots
- Communication Settings:
- Cable: Regular Ethernet cable

Adding a communication driver to your project

This section explains how to add a communication driver to your project.

1. On the **Insert** tab of the ribbon, in the **Communication** group, click **Add/Remove Driver**.
   The *Communication Drivers* dialog is displayed.



Communication Drivers dialog

2. In the *Available drivers* list, click the communication driver that you want to add.
3. Click **Select**.
   The driver is added to the *Selected drivers* list.
4. Click **OK**.
   The *Communication Drivers* dialog is closed and the selected driver is inserted in the **Drivers** folder in the Project Explorer.

About driver worksheets

Like the other parts of your project, communication with remote devices is controlled by worksheets. This section explains how to add worksheets to your project and then configure them to associate project tags with device registers.

Each selected driver includes a Main Driver Sheet (MDS) and one or more Standard Driver Sheets (SDS). The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. In contrast, Standard Driver Sheets can be inserted to define tag/ register associations that are triggered by specific project behaviors.

The configuration of these worksheets is described in detail in the "Communication" chapter of the *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only KAWR driver-specific parameters and procedures are discussed here.

Adding and configuring a Standard Driver Sheet

By default, a communication driver does not include any Standard Driver Sheets. This section explains how to add a Standard Driver Sheet to your project and then configure it.

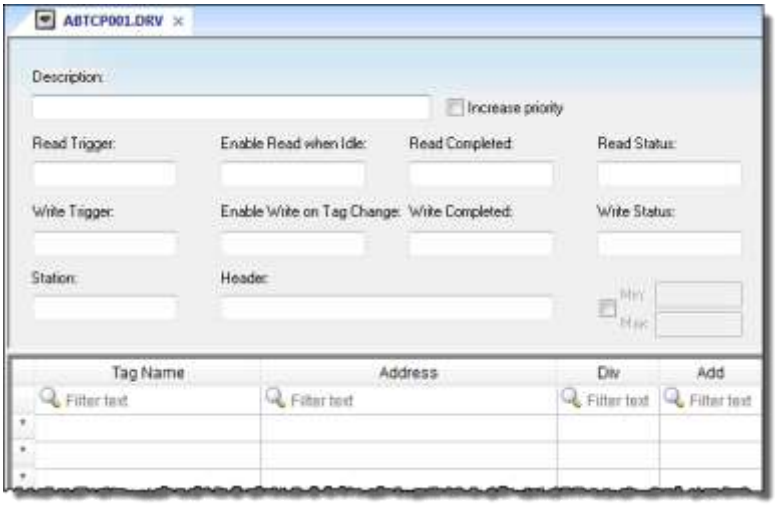
The KAWR driver must be added to the project before you can configure any of its worksheets. For more information, see Adding a communication driver to your project on page 6.

Standard Driver Sheets can be inserted to define additional tag/register associations that are triggered by specific project behaviors.

> **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the "Communication" chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the KAWR driver.

1. Do one of the following.
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **Driver Sheet** and then select **KAWR** from the list.
   - In the **Comm** tab of the Project Explorer, right-click the **KAWR** folder and click **Insert** on the shortcut menu.

   A new KAWR driver worksheet is inserted into the **KAWR** folder, and then it is automatically opened for configuring.



Standard Driver Sheet

> **Note:** Worksheets are numbered in order of creation, so the first worksheet is `KAWR001.drv`.

2. Configure the Station and Header fields as described below.

Station

Specify the station in the driver sheet using the following syntax. Note that this field cannot be left empty. Examples are given in the table below.

`Controller:IP Address:Port`

Where:

Controller is the controller type of the Kawasaki Robot. Valid types are D, E and S.

IP Address is the IP Address of the Robot on the Ethernet network.

Port is the port number to connect to the Robot.

You can also specify a tag in curly brackets to change the station during the runtime (e.g. `{Station}`), but the tag that is referenced must follow the same syntax and contain a valid value.

Station Formats

Examples
D:192.186.0.1:23
E:192.186.0.1:23

Header

Specify the item names for the command to be used. Refer to the AS Language Reference manual for the functionality of the commands. The header is given using the following syntax.

Header

Where:

Header is the item name.

Valid names for header are REAL, STRING, POSE, TRANSF, STATUS, SIGNAL, DREG, W1, W2, W3, W4, W5, W6, W10, W14.

The table below gives a description of the headers and their Read/Write attributes.

After you edit the **Header** field, the development application checks the syntax to determine if it is valid. If the syntax is invalid, then the development application automatically inserts the nearest matching command.

You can also specify an indirect tag (e.g. `{MyHeader}`), but the tag that is referenced must follow the same syntax and contain a valid value.

List of Supported Headers

Item Name (Header)	Description	Attribute
REAL	For Real Variables	Read/Write
STRING	For String Variables	Read/Write
POSE	Joint Position Variables	Read/Write
TRANSF	Transformation Position Variables	Read/Write
STATUS	Display status of current robot and system	Read
SIGNAL	I/O Internal Signals	Read all signals / Write only external and internal signals
DREG	Data Registers	Read/Write
W1	Joint Angle Value	Read
W2	Transformation Value	Read
W3	Joint Instruction Value	Read
W4	Joint Deviation	Read
W5	Encoded Value	Read
W6	Joint Speed	Read
W10	Motor Current Value	Read

W14	Motor Current Instruction Value	Read

3. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Address

Specify the variable name / signal or data register number that you want to communicate with. The syntax is as follows.

Name/Number.Joint Coordinate

Where:

Name/Number can any of the following.

- Variable Names for variable item headers like REAL, STRING,POSE (The name must start with a '#'), TRANSF
- I/O Internal Signal Numbers for the SIGNAL header
- Data Register numbers for the DREG header
- Status Command parameter names for the STATUS header
- Joint Values for the WHERE commands W1 to W6, W10 and W14.

Joint Coordinate

This is valid only for POSE and TRANSF headers. It is the joint or the transformation coordinates. Refer the table below for all the valid values.

Notes:

- SDS Block Size: The number of tags on a driver sheet for all headers is limited to 20 considering the length of the request being sent. This limitation is applied because of the limitation in the lenth of the request being sent in telnet command line.
- STRING length: When writing strings, the length of a single string being written might also be a limitation and it depends on the length accepted on the telnet command line. The total length of string combined with the tag name should not exceed 75.
- REAL Variables Accuracy: The REAL variables accuracy is 6 digits, this is a limitation on the protocol itself.

Note:

SIGNAL COMMAND USAGE :

When using SIGNAL command on the driver sheets, the addresses should be given such that all the addresses in one sheet fall under the same address group. For each signal command, 32 addresses can be retrieved and hence only addresses from the block of 32 should be given on each sheet. Here is a table showing different signal groups of addresses.

If the addresses are not given from same group, the results will be incorrect.

Addresses 1–32	Addresses 481–512
Addresses 33–64	Addresses 513–544
Addresses 65–96	Addresses 545–576
Addresses 97–128	Addresses 577–608
Addresses 129–160	Addresses 609–640
Addresses 161–192	Addresses 641–672
Addresses 193–224	Addresses 673–704
Addresses 225–256	Addresses 705–736
Addresses 257–288	Addresses 737–768
Addresses 289–320	Addresses 769–800
Addresses 321–352	Addresses 801–832
Addresses 353–384	Addresses 833–864
Addresses 385–416	Addresses 865–896
Addresses 417–448	Addresses 897–928
Addresses 449–480	Addresses 929–960

Similar address groups as listed above are used for Input and Internal signals which are in address ranges 1001–1960 and 2001–2960 respectively.

The following table shows the complete list of header, address formats, valid values for addresses, their attributes and data types along with examples on how to use them on the driver sheets.

Note:
The below table shows the possible response values for the STATUS command for each of its parameters.

STATUS VALUES

STATUS COMMAND PARAMETER	DATA TYPE	POSSIBLE VALUES
ERROR_STATUS	INT	1 if the value "During Error Condition" is present. 0 if it is not present.
OPERATING_MODE	INT	0 if "TEACH mode" is present 1 if "REPEAT mode" is present 2 if "REPEAT mode CYCLE START ON" is present-1 if none of the above are present.
MOTOR_POWER_STATE	INT	0 if "Motor power OFF" is present 1 if it is not present
MONITOR_SPEED	REAL	Value read from Robot
PROGRAM_ALWAYS_SPEED	REAL	Value read from Robot
PROGRAM_ACCURACY	REAL	Value read from Robot
PROGRAM_EXECUTION_STATE	INT	0 if "Program is not running" is present 1 if "Program running" is present -1 if neither of the above is present.
PROGRAM_COMPLETED_CYCL	ESINT	Value read from Robot
PROGRAM_REMAINING_CYCLE	S INT	Value read from Robot -1 if the value read from Robot is "Infinite"
EXECUTION_PROGRAM_NAME	STRING	Value read from Robot
EXECUTION_STEP_NUMBER	INT	Value read from Robot

Note: Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., "invalid block size") during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet.

For examples of how device registers are specified using **Header** and **Address**, see the following table.

Examples of Header and Address fields in Standard Driver Sheet

Item Type on Robot	Header on Studio	Address Format on Studio	Valid Address Values on Studio	Data Type	Attribute Read/Write (R/W)	Example on Studio
Program Variables	REAL	Variable Name	Variable Name	REAL	R/W	c_act[2,2] TestReal
	STRING	Variable Name	Variable Name	STRING	R/W	$String1 TestStr
	POSE	Variable Name. Joint Value	Variable Name. Valid Joint Values are JT1, JT2, JT3, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R/W	#pose1.JT1 #pose1.JT10 #pose2.JT5 #ABC.JT3
	TRANSF	Variable Name. Transf Value	Variable Name. Valid Transf Values are X, Y, Z, O, A, T, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R/W	transf.X transf.Y TestVar.A addr.JT8
Signal Values	SIGNAL	Output Signal Number	1 - 960	BOOL	R/W	1 500
		Input Signal Number	1001-1960	BOOL	R	1050 1100
		Internal Signal Numer	2001-2960	BOOL	R/W	2032 2050

For more information about the device registers and addressing, please consult the manufacturer's documentation.

4. Save and close the worksheet.

Configuring the Main Driver Sheet

When you add the KAWR driver to your project, the Main Driver Sheet is automatically included in the **KAWR** folder in the Project Explorer. This section describes how to configure the worksheet.

The KAWR driver must be added to the project before you can configure any of its worksheets. For more information, see Adding a communication driver to your project on page 6.

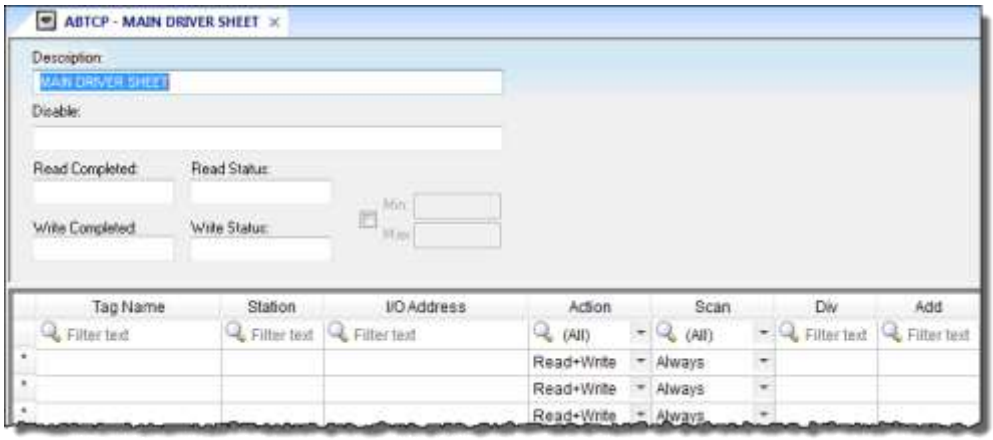
The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. The worksheet is continuously processed during project runtime.

Item Type on Robot	Header on Studio	Address Format on Studio	Valid Address Values on Studio	Data Type	Attribute Read/Write (R/W)	Example on Studio
Program Status	STATUS	Parameter Name	ERROR_STATUS	INT	R	ERROR_STATUS
		Parameter Name	OPERATING_MODE	INT	R	OPERATING_MODE
		Parameter Name	MOTOR_POWER_STATE	INT	R	MOTOR_POWER_STATE
		Parameter Name	MONITOR_SPEED	REAL	R	MONITOR_SPEED
		Parameter Name	PROGRAM_ALWAYS_SPEED	REAL	R	PROGRAM_ALWAYS_SPEED
		Parameter Name	PROGRAM_ACCURACY	REAL	R	PROGRAM_ACCURACY
		Parameter Name	PROGRAM_EXECUTION_STATE	INT	R	PROGRAM_EXECUTION_STATE
		Parameter Name	PROGRAM_COMPLETED_CYCLES	INT	R	PROGRAM_COMPLETED_CYCLES
		Parameter Name	PROGRAM_REMAINING_CYCLES	INT	R	PROGRAM_REMAINING_CYCLES
		Parameter Name	EXECUTION_PROGRAM_NAME	STRING	R	EXECUTION_PROGRAM_NAME
		Parameter Name	EXECUTION_STEP_NUMBER	INT	R	EXECUTION_STEP_NUMBER
Data Registers	DREG	Register Number	0-2047	INT	R/W	2, 345, 500
WHERE Commands	W1	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT1 JT4
	W2	Transf Coordinates	X, Y, Z, O, A, T, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	Y T JT10
	W3	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT2 JT6
	W4	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT4 JT5
	W5	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT1 JT3
	W6	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT6 JT1
	W10	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT2 JT14
	W14	Joint Coordinates	JT1, JT2, JT2, JT4, JT5, JT6, JT7, JT8, JT9, JT10, JT11, JT12, JT13, JT14, JT15, JT16	REAL	R	JT3 JT6

> **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the "Communication" chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the KAWR driver.

1. Do one of the following.
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **Main Driver Sheet** and then select **KAWR** from the list.
   - In the **Comm** tab of the Project Explorer, expand the **KAWR** folder and then double-click **MAIN DRIVER SHEET**.

   The Main Driver Sheet is displayed.



Main Driver Sheet

2. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Station

Please see station for standard driver sheet.

I/O Address

Specify the command item name and the variable name/signal or register number and the joint value if required (for POSE and TRANSF headers). The POSE header should always start with '#'.

The I/O address is given as in the following syntax.

Header*:*Name/Number*.*Joint Coordinate

Where:

Header is the item name for the command to be used.

Valid names for header are REAL, STRING, POSE, TRANSF, STATUS, SIGNAL, DREG, W1, W2, W3, W4, W5, W6, W10, W14.

Name/Number can be any of the following

- Variable Names for variable item headers like REAL, STRING, POSE, TRANSF
- IO/Internal Signal Number for the Signals header
- Data Register Number for the DREG header
- Status Command item names for the STATUS header
- Joint Values for the WHERE commands W1 to W6, W10 and W14.

Joint Coordinate is valid only for POSE and TRANSF headers. It is the joint or the transformation value. Refer the table of examples for Headers and Addresses in the section above for Standard Driver Sheet.

Refer to the table in standard driver sheet section for a complete list of valid address values, signal and data registers numbers and joint coordinates.

Examples of I/O Addresses on MDS

I/O Address	Example
REAL : Varname	REAL : c_act[2,2] REAL : Test
STRING : Varname	STRING : $String1 STRING : TextStr
POSE : Varname . JointValue	POSE : #pose1.JT1 POSE : #otherPose.JT4
TRANSF : Varname . TransfValue	TRANSF : ABCPose.X TRANSF : ABCPose.JT8
STATUS : ParameterName	STATUS : PROGRAM_ACCURACY STATUS : OPERATING_MODE
SIGNAL : Signal Number	SIGNAL : 10 SIGNAL : 1200 SIGNAL : 2002
DREG : Register Number	DREG : 20 DREG : 1000
W1 : JointValue	W1 : JT1 W1 : JT4
W2 : TransfValue	W2 : A W2 : JT14
W3 : JointValue	W3 : JT3 W3 : JT4
W4 : JointValue	W4 : JT5 W4 : JT6
W5 : JointValue	W5 : JT1 W5 : JT3

W6 : JointValue	W6 : JT2 W6: JT3
W10 : JointValue	W10 : JT1 W10 : JT2
W14 : JointValue	W14 : JT4 W14 : JT5

Note: NAMING CONVENTION FOR POSE AND TRANSF HEADERS:
On the Kawasaki Robot, the variable names for POSE and TRANSFORMATION are differentiated by prefixing a '#' sign for all POSE variables and the same convention is followed on Studio. Any variable name given in the address for the POSE header, will be prefixed by a '#' sign. If any variable name given in the address for the TRANSF header starts with '#' sign, the '#' will be removed.

Note: The Main Driver Sheet can have up to 32767 rows. If you need to configure more than 32767 communication addresses, then either configure additional Standard Driver Sheets or create additional instances of the driver.

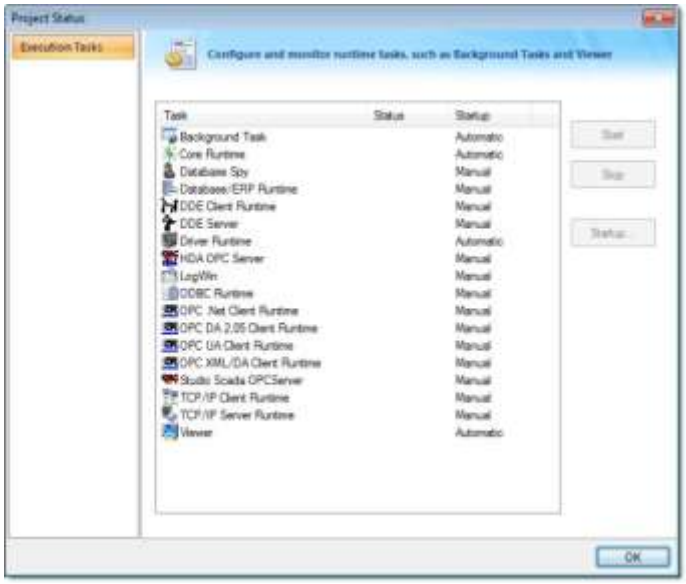
3. Save and close the worksheet.

Checking the Driver Runtime task

This section describes how to check the status of the Driver Runtime task in the list of execution tasks.

The Driver Runtime task handles communication with remote devices and the processing of the driver worksheets. By default, the task is configured to start up automatically when the project is run, but you can check it for yourself.

1. On the **Home** tab of the ribbon, in either the **Local Management** or the **Remote Management** group (depending on where you project server will be running), click **Tasks**.
   The *Project Status* dialog is displayed.



Project Status dialog

2. Verify that the **Driver Runtime** task is set to **Automatic**.
   - If the setting is correct, then proceed to the next step.
   - If the **Driver Runtime** task is set to **Manual**, select the task and then click **Startup** to change the task to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.

Troubleshooting

This section lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.

Checking status codes

If the KAWR driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** and **Write Status** fields of the driver sheets will receive a status code. Use this status code and the following tables to identify what kind of failure occurred and how it might be resolved.

Status codes for the driver

Error	Description	Possible Causes	Procedure To Solve
1	Illegal WRITE. This is a Read-Only command	The header that is attempted to write is ReadOnly. Commands like STATUS, WHERE commands, Input Signals are all read-only.	Check the Table in Standard Driver Sheets for the read/ write attributes of each command item
2	Failed to Initialize TCP Connection	Unable to initialize a TCP connection	Check the station and network connections and try again
3	Failed to Initialize Terminal	Unable to initialize the Terminal on the Robot	Check the station and connections and restart the driver.
4	Failed to Send Terminal Type	Error when sending the Terminal type to the Robot	Check the station and connections and restart the driver.
5	Login Failed	The Login failed on the Robot, station parameters might be incorrect	Check the station, controller type given and try again.
6	Error in Sending Request	The headers or addresses might be incorrect.	Check the valid table in Standard Driver Sheet above for a list of valid headers and addresses and make sure they are configured correctly.
7	Error in Response from Robot	The Robot sent an error in the response. One or more parameters in the command sent by the driver was incorrect or the variable name given does not exist. The AS Language reference manual has a complete listing of these error codes.	The error sent in the response is displayed in the previous line. Check the error, correct the parameters configured on the driver and try again.

Common status codes

Status Code	Description	Possible Causes	Procedure To Solve
0	OK	Communicating without error.	None required.
-15	Timeout waiting for message to start	• Disconnected cables. • PLC is turned off, in stop mode, or in error mode. • Wrong station number. • Wrong parity (for serial communication). • Wrong RTS/CTS configuration (for serial communication).	• Check cable wiring. • Check the PLC mode — it must be RUN. • Check the station number. • Increase the timeout in the driver's advanced settings. • Check the RTS/CTS configuration (for serial communication).
-33	Invalid driver configuration file	The driver configuration file (*drivername*.INI) is missing or corrupt.	Reinstall the driver.

-34	Invalid address	The specified address is invalid or out of range.	Check the supported range of addresses described in this document, and then correct the address.
-35	Driver API not initialized	The driver library was not initialized by the driver.	Contact technical support.
-36	Invalid data type	The specified data type is invalid or out of range.	Check the supported data types described in this document, and then correct the data type.
-37	Invalid header	The specified header in the driver worksheet is invalid or out of range.	Check the supported range of headers described in this document, and then correct the header.
-38	Invalid station	The specified station in the driver worksheet is invalid or out of range.	Check the supported station formats and parameters described in this document, and then correct the station.
-39	Invalid block size	Worksheet is configured with a range of addresses greater than the maximum block size.	Check the maximum block size number of registers described in this document, and then configure your driver worksheet to stay within that limit. Keep in mind that you can create additional worksheets. **Note:** If you receive this error from a Main Driver Sheet or Tag Integration configuration, please contact Technical Support.
-40	Invalid bit write	Writing to a bit using the attempted action is not supported.	• Writing to a bit using Write Trigger is not supported in some drivers. Modify the driver worksheet to use Write On Tag Change. • The bit is read-only.
-42	Invalid bit number	The bit number specified in the address is invalid. The limit for the bit number depends on the registry type.	Check the addresses to see if there are bit numbers configured outside the valid range for the registry.
-43	Invalid byte number	The byte number specified in the address is invalid. The limit for the byte number depends on the registry type.	Check the addresses to see if there are byte numbers configured outside the valid range for the registry.
-44	Invalid byte write	Writing to a byte using the attempted action is not supported.	The byte is read-only or inacessible.
-45	Invalid string size	The string is more than 1024 characters.	Modify the addresses that have string data type to be less than 1024 characters.
-56	Invalid connection handle	The connection is no longer valid.	Please contact Technical Support.
-57	Message could not be sent	The socket was unable to send the TCP or UDP message.	• Check the station IP address and port number. • Confirm that the device is active and accessible. Try to ping the address.
-58	TCP/IP could not send all bytes	The TCP/IP stack was not able to send all bytes to destination.	• Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.

-60	Error to establish TCP/IP connection	Error while establishing a TCP/IP connection with the slave device. Possibly incorrect IP address or port number in the specified station.	• Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-61	TCP/IP socket error	The TCP/IP connection has been closed by the device.	Confirm that the device is active and accessible. Try to ping the address.
-62	UDP/IP receive call returned error	The UDP socket is in error.	• Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-63	UDP/IP error initializing	The UDP socket initialization failed.	Confirm that the operating system supports UDP sockets.
-64	UDP/IP receive call returned error	The UDP socket is in error.	• Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-65	UDP/IP bind error, port number may already be in use	The driver was not able to bind the UDP port.	• Check the port number used by the driver. • Check for other programs that might be bound to the UDP port.

Monitoring device communications

You can monitor communication status by establishing an event log in Studio's *Output* window (LogWin module). To establish a log for Field Read Commands, Field Write Commands and Serial Communication, right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the LogWin module to establish an event log on a remote unit that runs Windows Embedded. The log is saved on the unit in the `celog.txt` file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

Contacting Technical Support

If you must contact Technical Support, please have the following information ready:

- **Operating System** and **Project Information**: To find this information, click **Support** in the **Help** tab of the ribbon.
- **Driver Version** and **Communication Log**: Displays in the *Output* window (LogWin module) when the driver is enabled and the project is running is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision history

This section provides a log of all changes made to the driver.

Revision history

Driver Version	Revision Date	Description of Changes	Author
1.0	Oct 12, 2015	• First driver revision	Priya Yennam
1.1	Nov 09, 2015	• Improved functionality of STRINGS, SIGNAL headers and updated documentation	Anushree Phanse
1.2	Nov 13, 2015	• Fixed OPERATING_MODE status and updated documentation	Eduardo Castro
1.3	June 28, 2018	• Fixed issues with telnet message processing, reset, disconnect and reconnect issues with the machine	Anushree Phanse