XRCM Communication Driver

Master Driver for Serial Communication
with Motoman XRC Robot Controllers

Contents

Introduction

The XRCM driver enables communication between the Studio system and Motoman XRC Robot Controllers using their BSC-compliant protocol via RS 232, according to the specifications discussed in this document.

This document will help you to select, configure and execute the XRCM driver, and it is organized as follows:

- **Introduction**: This section, which provides an overview of the document.
- **General Information**: Identifies all of the hardware and software components required to implement communication between the Studio system and the target device.
- **Selecting the Driver**: Explains how to select the XRCM driver in the Studio system.
- **Configuring the Driver**: Explains how to configure the XRCM driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the XRCM driver during application runtime.
- **Troubleshooting**: Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **Revision History**: Provides a log of all changes made to the driver and this documentation.

✎ **Notes:**

- This document assumes that you have read the "Development Environment" chapter in Studio's *Technical Reference Manual*.
- This document also assumes that you are familiar with the Microsoft Windows NT/2000/XP environment. If you are not familiar with Windows, then we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter identifies all of the hardware and software components required to implement serial communication between the XRCM driver in Studio and a target device. The information is organized into the following sections:

- Device Specifications
- Network Specifications
- Driver Characteristics
- Conformance Testing

Device Specifications

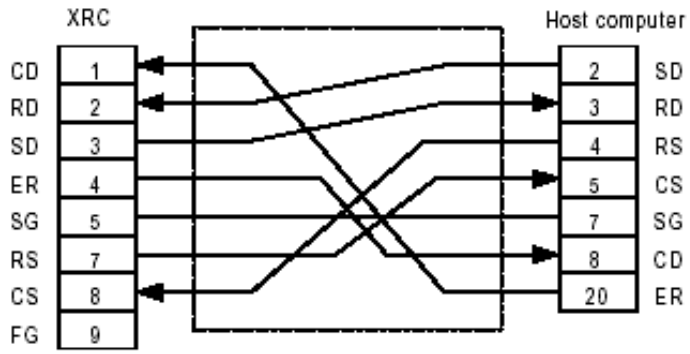
To establish communication, your target device must meet the following specifications:

- **Manufacturer**: Motoman
  - **Compatible Equipment**: XRC Robot Controller

Network Specifications

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

- **Device Communication Port**: RS232 Port
- **Physical Protocol**: RS232/RS485
- **Logic Protocol**: BSC-compliant Master Protocol
- **Device Runtime Software**: None
- **Specific PC Board**: None
- **Cable Wiring Scheme**: We can see it on the following picture bellow:



Driver Characteristics

The XRCM driver package consists of the following files, which are automatically installed in the **`\DRV`** subdirectory of Studio:

- **`XRCM.INI:`** Internal driver file. *You must not modify this file*.
- **`XRCM.MSG:`** Internal driver file containing error messages for each code. *You must not modify this file*.
- **`XRCM.PDF:`** This document, which provides detailed information about the XRCM driver.
- **`XRCM.DLL:`** Compiled driver.

> **Note:**
> You must use Adobe Acrobat® Reader™ to view the **`XRCM.PDF`** document. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's Web site.

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

- Windows NT/2000/XP

The XRCM driver supports the following registers:

Register Type	Read	Write
IO or I/O	•	•
RA_XYZ	•	•
RESET	•	•
CANCEL	•	•
SVON	•	•
BYTE	•	•
INTEGER	•	•
DOUBLE	•	•
REAL	•	•
ROBOT_AXIS	•	•
BASE_AXIS	•	•
STATION_AXIS	•	•
RSTATS	•	–
RJSEQ	•	–
HOLD	•	•
RALARM	•	–
JSEQ	–	•
START	–	•
MODE	–	•
SETMJ	–	•
CYCLE	–	•

Conformance Testing

The following hardware/software was used for conformance testing:

Configuration:

– **Baud Rate**: 9600
– **Protocol**: BSC-compiliant Master Protocol
– **Data Bits**: 8
– **Stop Bits**: 1
– **COM Port**: COM1
– **Parity**: None

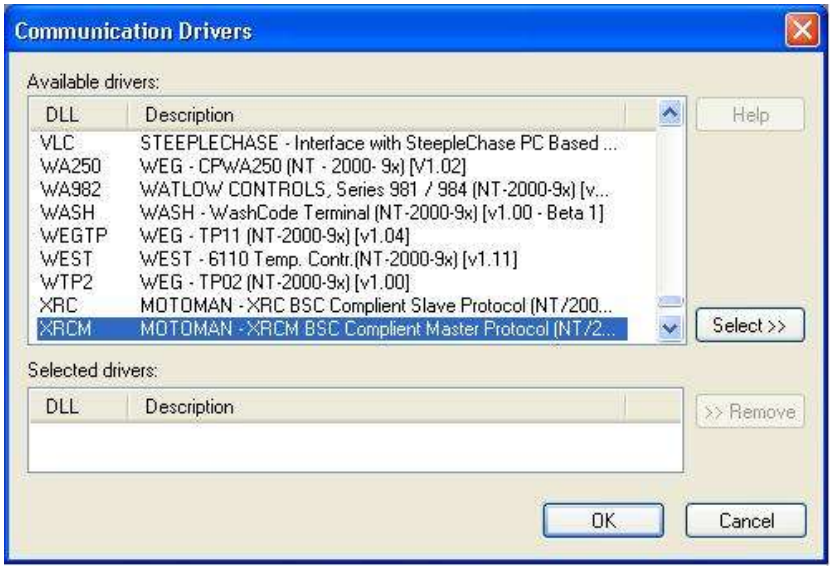
Cable: Use specifications described in the “Network Specifications” section above.

Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
1.00	6.1	Windows XP	Windows XP Windows CE 4.0	PLC AEG CPU 984L

Selecting the Driver

When you install Studio, all of the communication drivers are automatically installed in the **\DRV** subdirectory but they remain dormant until manually selected for specific applications. To select the XRCM driver for your Studio application:

1. From the main menu bar, select **Insert** → **Driver** to open the *Communication Drivers* dialog.
2. Select the **XRCM** driver from the *Available Drivers* list, and then click the **Select** button.



Communication Drivers Dialog

3. When the **XRCM** driver is displayed in the **Selected Drivers** list, click the **OK** button to close the dialog. The driver is added to the *Drivers* folder, in the *Comm* tab of the Workspace.

> **Note:**
> It is not necessary to install any other software on your computer to enable communication between Studio and your target device.

> **Attention:**
> For safety reasons, you must take special precautions when installing any physical hardware. Please consult the manufacturer's documentation for specific instructions.

Configuring the Driver

Once you have selected the XRCM driver in Studio, you must properly configure it to communicate with your target device. First, you must set the driver's communication settings to match the parameters set on the device. Then, you must build driver worksheets to associate database tags in your Studio application with the appropriate addresses (registers) on the device.

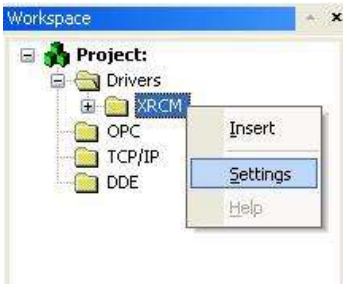
Configuring the Communication Settings

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

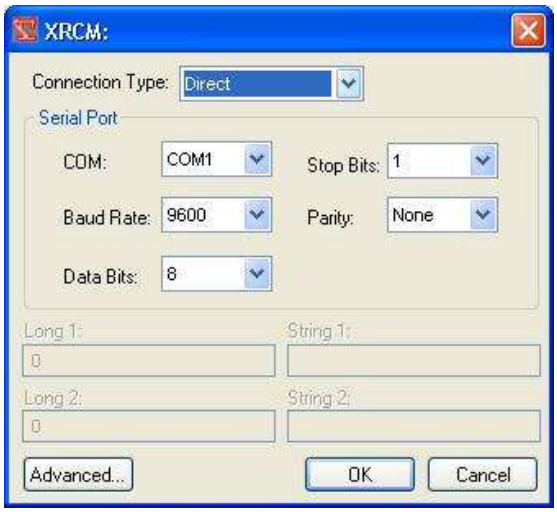
For the purposes of this document, only XRCM driver-specific settings and procedures will be discussed here. To configure the communication settings for the XRCM driver:

1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The XRCM driver is listed here as a subfolder.

2. Right-click on the XRCM subfolder and then select the **Settings** option from the pop-up menu. The *XRCM: Communication Settings* dialog is displayed:



Select Settings from the Pop-Up Menu



XRCM: Communication Settings Dialog

3. In the *Communication Settings* dialog, configure the driver settings to enable communication with your target device. To ensure error-free communication, the driver settings must *exactly match* the corresponding settings on the device. Please consult the manufacturer's documentation for instructions how to configure the device and for complete descriptions of the settings.

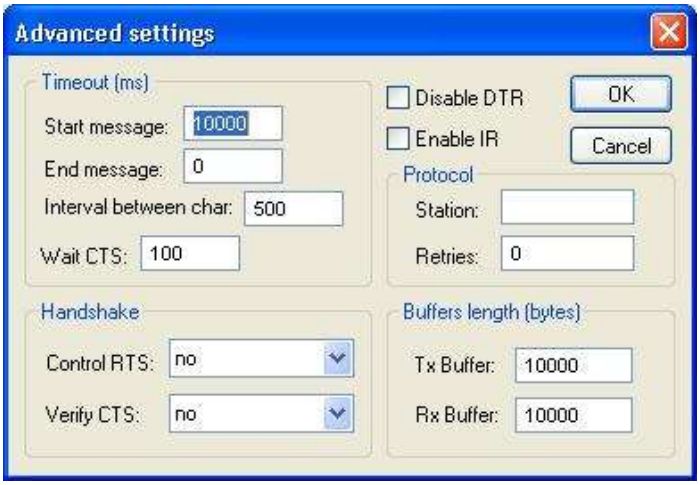
   Depending on your circumstances, you may need to configure the driver *before* you have configured your target device. If this is the case, then take note of the driver settings and have them ready when you later configure the device.

> ➲ **Attention:**
>
> For safety reasons, you **must** take special precautions when connecting and configuring new equipment. Please consult the manufacturer's documentation for specific instructions.

The communication settings and their possible values are described in the following table:

Parameters	Default Value	Valid Values	Description
COM	`COM1`	`COM1` to `COM8`	Serial port of the PC used to communication with the device.
Baud Rate	`9600`	`110` to `57600`	Communication rate of data, in bits per second (bps).
Data Bits	`8`	`5` to `8`	Number of data bits used in the protocol.
Stop Bits	`1`	`1` or `2`	Number of stop bits used in the protocol.
Parity	`None`	`even`, `odd`, `none`, `space` or `mark`	Parity of the protocol.

4. If you are using a Data Communication Equipment (DCE) converter (e.g., 232/485) between your PC and your target device, then you must also adjust the **Control RTS** (Request to Send) setting to account for the converter. In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



Advanced Settings Dialog

When the dialog is displayed, configure the **Control RTS** setting using the following information:

Setting	Default	Values	Description
Control RTS	`no`	`no`	Do not set the RTS (Request to Send) handshake signal. IMPORTANT: If you are using Windows 95/98 or Windows CE with the correct RS232/RS485 adapter (i.e., without RTS control), then you must select this option.
		`yes`	Set the RTS (Request to Send) handshake signal before communication. IMPORTANT: If you are using Windows NT and the Cutler-Hammer RS232/RS485 adapter, then you must select this option.
		`yes+echo`	Set the RTS (Request to Send) handshake signal before communication, and echo the signal received from the target device.

You do not need to change any other advanced settings at this time. You can consult the Studio *Technical Reference Manual* later for more information about configuring these settings.

5. Click **OK** to close the *Advanced Settings* dialog, and then click **OK** to close the *Communication Settings* dialog.

Configuring the Driver Worksheets

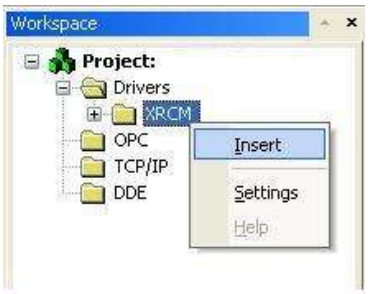
A selected driver includes one or more driver worksheets, which are used to associate database tags in Studio with registers on the target device. Each worksheet is triggered by specific application behavior, so that the tags / registers defined on that worksheet are scanned only when necessary – that is, only when the application is doing something that requires reading from or writing to those specific tags / registers. Doing this optimizes communication and improves system performance.

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

> **Note:**
> We recommend configuring device registers in sequential blocks in order to maximize performance.

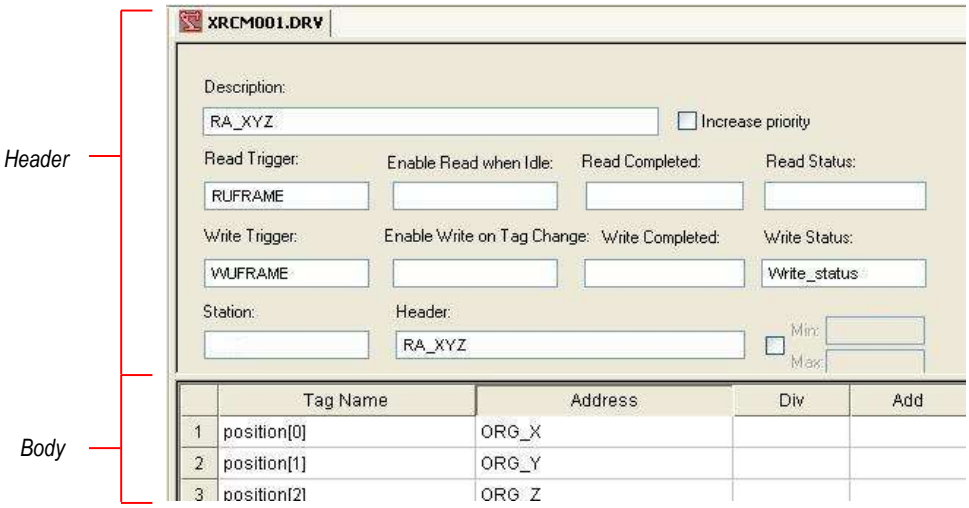
To insert a new driver worksheet:

1. In the *Comm* tab, open the *Drivers* folder and locate the *XRCM* subfolder.
2. Right-click on the *XRCM* subfolder, and then select **Insert** from the pop-up menu:



Inserting a New Worksheet

A new XRCM driver worksheet is inserted into the XRCM subfolder, and the worksheet is opened for configuration:



XRCM Driver Worksheet

Most of the fields on this worksheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station**, **Header**, and **Address** fields use syntax that is specific to the XRCM driver.

3. Configure the **Station** and **Header** fields as follows:

   - **Station** field: Not used by this driver; the driver will communicate directly with whatever station is connected via the COM port configured in the *Communication Settings* dialogue (see above).

   - **Header** field: Specify the address of the first register of a block of registers on the target device. The addresses declared in the *Body* of the worksheet are simply offsets of this **Header** address. When Read/Write operations are executed for the entire worksheet (see **Read Trigger** and **Write Trigger** above), it scans the entire block of registers from the first address to the last.

     The **Header** field uses the following syntax:

     - For **`IO and I/O`** (Input/Output) registers *only*, use the following syntax:

       `<Type>:<Address Reference>`

       Example — **`IO:2`**

     - For **`RA_XYZ`** registers *only*, use the following syntax:

       `<Type>:<User Coordinate>`

       Example — **`RA_XYZ:2`**

     - For all other registers, use the following syntax:

       `<Type>`

       Example — **`BYTE`**

     Where:

     - ***`<Type>`*** is the command type: **`IO`** (or **`I/O`**), **`RA_XYZ`**, **`RESET`**, **`CANCEL`**, **`SVON`**, **`BYTE`**, **`INTEGER`**, **`DOUBLE`**, **`REAL`**, **`ROBOT_AXIS`**, **`BASE_AXIS`**, **`STATION_AXIS`**, **`RSTATS`**, **`RJSEQ`**, **`HOLD`** or **`RALARM`**. See the table on the next page for descriptions of these commands.

     – ***`<Address Reference>`*** is the initial address (reference) of the configured type.

     – ***`<User Coordinate>`*** `is the user coordinate number in Robot Axis.`

     0 : Reserved
     1 : Reserved
     2 : User coordinate 1…
     25 : User coordinate 24

     After you edit the **Header** field, Studio checks the syntax to determine if it is valid.

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

> ⛔ **Attention:**
> You cannot leave the **Header** field blank; you must specify some values.

The following table describes the action performed by each command:

Command	Action
IO or **I/O**	Read out or Write in (change) I/O signal status using the host computer.
RA_XYZ	Reads / Writes a specified user coordinate.
RESET	Resets an alarm of manipulator.
CANCEL	Cancels an error.
SVON	Turns servo power supply ON/OFF.
BYTE	Receives variable data from a host computer and write it in a specified variable. Sends variable data to a host computer.
INTEGER	
DOUBLE	
REAL	
ROBOT_AXIS	Receives variable data from a host computer and write it in a specified variable. Sends variable data to a host computer.
BASE_AXIS	
STATION_AXIS	
RSTATS	Reads the status of mode, cycle, operation, alarm error, and servo.
RJSEQ	Reads the current job name, line number and step number.
HOLD	Turns HOLD on/off.
RALARM	Reads the error alarm code.
JSEQ	Sets a job name and a line number.
START	Starts a job.
MODE	Selects a mode.
SETMJ	Sets a specified job as a master job.
CYCLE	Selects cycle.

4. The **Address** field contains the number of positions to be read or write.

For examples of valid addresses for each command type (as the command is configured in the **Header** field), see the following table:

Header Field	Address Field	Description	Comment
IO	**<Base>.<Bit>** Where Base ranges from **0** to **8220** and Bit ranges from **0** to **7**. The base address is the IO address divided by 10. In the robot, you'll see YYYYX, where YYYY indicates the base address and X the bit. **Example**: To read IO 30025, use 3002.5.	Read and write I/Os.	
RA_XYZ:0	**ORG_X**	ORG coordinate on X axis	▪ `All of them must be configured.` ▪ ORG, XX, XY coordinates are read in the base coordinate system. ▪ In a system having no external axis, Data-23 to Data-28 are "0". ▪ If the specified user coordinate system is not registered, an error occurs. ▪ If ORG, XX, and XY have different base axis data, an error occurs.
	ORG_Y	ORG coordinate on Y axis	
	ORG_Z	ORG coordinate on Z axis	
	ORG_TX	ORG wrist angle TX	
	ORG_TY	ORG wrist angle TY	
	ORG_TZ	ORG wrist angle TZ	
	ORG_TYPE	ORG type	
	XX_X	XX coordinate on X axis	
	XX_Y	XX coordinate on Y axis	
	XX_Z	XX coordinate on Z axis	
	XX_TX	XX wrist angle TX	
	XX_TY	XX wrist angle TY	
	XX_TZ	XX wrist angle TZ	
	XX_TYPE	XX type	
	XY_X	XY coordinate on X axis	
	XY_Y	XY coordinate on Y axis	
	XY_Z	XY coordinate on Z axis	
	XY_TX	XY wrist angle TX	
	XY_TY	XY wrist angle TY	
	XY_TZ	XY wrist angle TZ	
	XY_TYPE	XY type	
RA_XYZ:0	**TOOL**	(0 to 23)	
	PULSE_7	Number of 7th axis pulses	
	PULSE_8	Number of 8th axis pulses	

Header Field	Address Field	Description	Comment
	PULSE_9	Number of 9th axis pulses	
	PULSE_10	Number of 10th axis pulses	
	PULSE_11	Number of 11th axis pulses	
	PULSE_12	Number of 12th axis pulses	
RESET	—	—	These commands do not use the **Address** field. It is not necessary to enter any value.
CANCEL			
SVON	Any value	You must configure a tag.	Tag value: ▪ **0** : OFF ▪ **1** : ON
BYTE	**0 to 90000**	Variable No	
INTEGER			
DOUBLE			
REAL			
ROBOT_AXIS	**0** to **10**	Configure tags to hold the position information.	Configure: 8 addresses – Pulse or 10 addresses -- Cartesian Representing: Coordinate values (x, y and z), Wrist angle (Rx, Ry and Rz), Form and Tool
BASE_AXIS			
STATION_AXIS			
RSTATS	Any two addresses	You must configure two tags.	Response format requires two points of data.
RJSEQ	Any three addresses	You must configure three tags.	Response format requires three points of data. The first address is a string, and contains the job name, the second contains the line number, and the third contains the step number.
HOLD	Any value	You must configure a tag.	Tag Value: ▪ 0 : OFF ▪ 1 : ON
RALARM	Any nine addresses	You must configure nine tags.	Response format requires nine points of data.
JSEQ	Command requires two points of data. The first address is a string, and contains the job name and the second contains the line number.	You must configure two tags.	
START	Command requires one Address Field. The value is a string, and contains the job name or is empty.	If a job name is specified for an operand, the relation between the job and the master job is checked and the execution is started from the beginning of the job.	If an empty string is set in the tag, the command is sent without parameters.

Header Field	Address Field	Description	Comment
MODE	Mode number. = 1	Teach mode	
	Mode number. = 2	Play mode	
SETMJ	Command requires one Address Field. The value is a string, and contains the job name.	At the same time, the specified job is set as execution job.	
CYCLE	Cycle-No = 1.	Step	
	Cycle-No = 2.	1 cycle	
	Cycle-No = 3.	Auto	

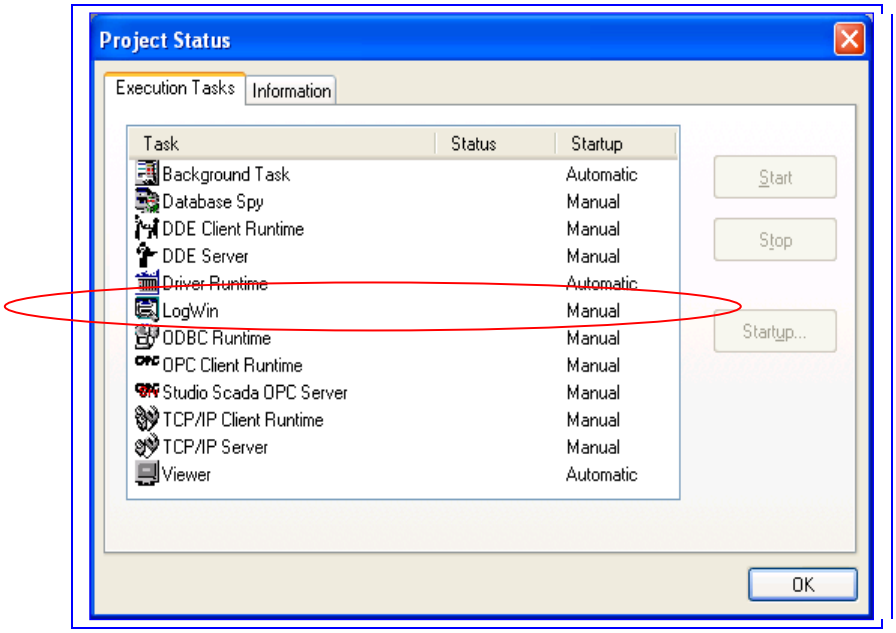
> **Note:**
> For more information about registers and addressing, please consult the manufacturer's documentation.

Executing the Driver

By default, Studio will automatically execute your selected communication driver(s) during application runtime. However, you may verify your application's runtime execution settings by checking the *Project Status* dialog.

To verify that the communication driver(s) will execute correctly:

1. From the main menu bar, select **Project** → **Status**. The *Project Status* dialog displays:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
   - If the setting is correct, then proceed to step 3 below.
   - If the **Driver Runtime** task is set to **Manual**, then select the task and click the **Startup** button to toggle the task's *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

Troubleshooting

If the XRCM driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** or **Write Status** fields of the Driver Sheet will receive an error code. Use this error code and the following table to identify what kind of failure occurred.

Error	Description	Possible Causes	Procedure to Solve
0	OK	Communication without problems.	None required.
1	Error waiting ACK0	Error occurred in Serial communication.	Check the Serial communication settings both on the target device and in Studio.
2	Error waiting ACK1		
3	Error waiting ENQ		
4	Error waiting SOH		
5	Error waiting EOT		
7	Invalid message	Error using command.	Check the Driver Sheet configuration.
8	Error Initialize	Error occurred while initializing the communication driver.	Check the Serial communication settings both on the target device and in Studio.
9	Invalid Read	Error occurred while reading or writing a specific command.	Check the individual commands configured in the Driver Sheet.
10	Invalid Write		
11	Error Read Finish		
12	Error Write Finish		
13	Invalid Address Coordinate	Error using address coordinates.	Check for valid address configuration.
14	Invalid Read Command	Specific Read or Write command is not allowed by the device.	Check the individual commands configured in the Driver Sheet.
15	Invalid Write Command		
16	Buffer full	Too many commands configured in a single Driver Sheet.	Review the Driver Sheet configuration. Eliminate unnecessary commands or divide the commands into additional Driver Sheets.
17	Invalid Block Size	Driver Sheet configuration is not correct.	Check correct configuration.

⇨ **Tip:**

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 (**Tools** → **LogWin**) to establish an event log on a remote unit that runs Windows CE. 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.

If you need to contact us for technical support, then please have the following information available:

- **Operating System** (type and version): To find this information, select **Tools** → **System Information**.
- **Project Information**: To find this information, select **Project** → **Status.**
- **Driver Version** and **Communication Log**: Displays in the Studio *Output* window when the driver is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Graziane C. Forti	22 Dec 2006	Initial release.
B	1.00	Michael D. Hayden	30 Jan 2007	Minor editing and formatting.
C	1.1	André Körbes Ajay Samrat	31 May 2012	Fixed RJSEQ and IO issues. Improved documentation about BYTE, INTEGER, DOUBLE and REAL. Included JSEQ, START, MODE, SETMJ and CYCLE commands.