

Memory Card Usability on Logix5000 Controllers

Catalog Numbers CompactLogix 5370, ControlLogix 5560, ControlLogix 5570, ControlLogix 5580



Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.



WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.



SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.



BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.



ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Summary of Changes	5
Preface	
Purpose of This Manual	7
Terminology	7
Additional Resources	7
Chapter 1	
Introduction	9
Memory Card Enhancements	9
Sample File Compatibility	11
Memory Card File System Features	11
Possible Applications of the Memory Card File System	12
Chapter 2	
Introduction	13
Restrictions of the File System	13
Number of Open Files	13
Maximum Data Per Message	13
Removal and Insertion of Memory Card	13
Minimum File Size/Resolution	14
Memory Resources and Performance	14
Controllers Supported	14
Limited Life	14
Read or Write Portions of a File	14
Number of Files that Can Be Created	14
Chapter 3	
Introduction	15
CF_Data_Working Tag	16
Create Command	17
Set Up the Create Command	18
Write Command	18
Set Up the Write Command	19
Read Command	19
Set Up the Read Command	20
Delete Command	20
Set Up the Delete Command	20
Determine the Number of Bytes to Read or Write	21
Chapter 4	
Introduction	23
Requirements	23
Configure Message Instructions	24
Set the Controller Slot Number	24
Create a File on the Memory Card	25

	Write Data to the Memory Card	28
	Read Data from the Memory Card.....	30
	Delete a File on the Memory Card	32
	Build an Example.....	33
	Set Up Recipe Data	34
	Create the File.....	36
	Write the Recipe	37
	Read the Recipe	37
	Delete the Recipe.....	38
Format a Memory Card	Appendix A	
	Introduction.....	39
	Requirements	39
	Format with a Personal Computer and a Memory Card Reader ...	39
	Format a CompactFlash Card with RSLogix 5000 Software.....	41
	Format an SD Card with the Logix Designer Application.....	45
Performance Data	Appendix B	
	Introduction.....	49
	Performance Data Tables	50
	Baseline Performance When Accessing 450 Bytes	50
	Baseline Performance When Accessing 4500 Bytes	50
Troubleshoot the Memory Card File System	Appendix C	
	Introduction.....	51
	Status User-defined Data Type	51
	Tag Layout.....	52
	Message Error Codes	53
Expand the Code to Support More Than 4500 Bytes of Data	Appendix D	
	Introduction.....	55
	Modify the Application to Access More Data	55
	Access 10,000 Bytes Example	56
	Modify the Arrays	57
	Modify the Data Element	57
Additional SD Card Resources	Appendix E	
	Appending Data	59
Index	61

This manual contains new and updated information as indicated in the following table

Topic	Page
General Secure Digital Card Use	Throughout
Sample File Compatibility	11
Format an SD Card with the Logix Designer Application	45
Performance Data Tables	50
Additional SD Card Resources	59

Notes:

Purpose of This Manual

This manual describes how you can use the CompactFlash file system or a Secure Digital card on Logix5000™ controllers. The manual uses the term ‘memory card’ to cover both CompactFlash cards and Secure Digital cards. The Logix5000 controller type that you use determines which type of memory card to use.

This document explains each feature and provides you with examples that illustrate how to use those features. After you read and understand this document and the examples, you will be able to add the Memory Card file-system functionality to any of your programs.

Terminology

Studio 5000 Logix Designer® application and RSLogix 5000® software are used to program specific Logix5000 controllers. The use of each programming tool is dependent on the controller being used. This manual uses the name of each tool where appropriate.

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, http://www.rockwellautomation.com/global/certification/overview.page	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at <http://www.rockwellautomation.com/global/literature-library/overview.page>. To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Notes:

Memory Card Enhancements Overview

Introduction

This chapter reviews the enhancements for the memory card file system for both CompactFlash cards and SD cards.

Topic	Page
Memory Card Enhancements	9
Sample File Compatibility	11
Memory Card File System Features	11
Possible Applications of the Memory Card File System	12

Memory Card Enhancements

CompactFlash and SD cards are available in two sizes. Base your selection on the amount of memory you need. [Table 1](#) lists four memory-card storage options that are based on capacity, controller, and software. We recommend using only Allen-Bradley® memory cards. The use of any third-party SD card that is not an official Allen-Bradley product is done so at your own risk.

Table 1 - Memory Card Types and Controller Compatibility

Compatible Logix5000™ Controllers	Memory Card Type and Catalog Number	Storage Capacity	Supported Software
ControlLogix® 5560	CompactFlash <ul style="list-style-type: none"> • 1784-CF 64 • 1784-CF128 	<ul style="list-style-type: none"> • 64 MB • 128 MB 	RSLogix 5000® software, version 11 or later
CompactLogix™ 5370	Secure Digital <ul style="list-style-type: none"> • 1784-SD1 • 1784-SD2 	<ul style="list-style-type: none"> • 1 GB • 2 GB 	RSLogix 5000 software, version 20 Logix Designer application, version 21 or later
ControlLogix 5570			RSLogix 5000 software, version 18 or later Logix Designer application, version 21 or later
ControlLogix 5580	Secure Digital <ul style="list-style-type: none"> • 1784-SD2 	<ul style="list-style-type: none"> • 2 GB 	Logix Designer application, version 28 or later

The memory card file system has gone through several enhancements. Versions 17...20 use CompactFlash cards and versions 21 and higher use SD cards.

Table 2 - Software Version Memory Card Features

Software Version	Memory Card Features
RSLogix 5000 Software Version 11	The CompactFlash card was only used to save project image files with the NVS feature.
RSLogix 5000 Software Version 12	The CompactFlash card stored firmware and project image files.
RSLogix 5000 Software Version 16	The CompactFlash card functionality has been extended to include a file system that can be used to store values, recipes, log data, and other user data.
RSLogix 5000 Software Version 18	The SD card stores multiple projects and associated firmware. The card also overwrites any project on the card with the same name, and it loads the most recent stored project. There are also two settings to the card: <ul style="list-style-type: none">• Unlocked: Leaves existing data and creates folders and files for the project and firmware.• Locked: Does not allow writing to the card

Sample File Compatibility

Table 3 represents the compatibility of sample projects.

Table 3 - Application Code Support

Logix5000™ Controllers	Compatibility	
	V3.002 ACD	V4.000 ACD
CompactLogix 5370 ControlLogix 5560 ControlLogix 5570	Yes	Yes
ControlLogix 5580	No	Yes

To format a CompactFlash card with RSLogix 5000 software, see page [39](#), and to see a similar step-by-step example with an SD card, see Appendix A, page [45](#).

Memory Card File System Features

The file system can perform various operations to read or write data to the memory card. The following are the basic operations:

- Create a file
- Write to a file
- Read from a file
- Delete a file

There is an additional command, verification if a memory card is present, but as a user you do not access this command. It runs as a background check.

For each of the basic operations of the file system, there are related functionalities:

- Open a file
- Close a file
- Set the file pointer
- Calculate file checksums
- Verify file checksums
- Data parsing
- Error handling

The application code is designed to handle a file up to 4500 bytes in size.

For more information on how to increase file size, refer to [Appendix D Expand the Code to Support More Than 4500 Bytes of Data](#).

Possible Applications of the Memory Card File System

A possible application would be to store a collection of recipes on the memory card. When an application runs a particular recipe, the recipe can be read from a file on the memory card. If changes were made to the recipe that are intended to be permanent, and are not minor adjustments that are made for fluid conditions, the revised recipe can be written back to the memory card for later use. If the data is not written back to the memory card, those changes are not saved permanently.

Memory cards can also be read and written to with a personal computer with the use of a memory card reader. However, with the methods that are described in this document, tag values are written out as binary data to the memory card.

A text editor opens the file, but the contents that are displayed are the ASCII equivalent of the binary data. The application does not allow files to be created on a personal computer and then read into the controller, because checksums are embedded into the data. If you create a file on the personal computer, it does not have the checksums embedded. When the data is read, it fails the checksum test.

With the RSLogix 5000 software, version 16.000 or later, and the application that is provided in this publication, you can read and write data to the memory card. This data could include tag values, recipes, or data log information, just to name a few. However, the programs only read and write files in the root directory.

Memory Card File System Restrictions

Introduction

This chapter describes the restrictions of the memory card file system.

Topic	Page
Restrictions of the File System	13
Number of Open Files	13
Maximum Data Per Message	13
Memory Resources and Performance	14
Limited Life	14
Read or Write Portions of a File	14

Restrictions of the File System

The following are the restrictions of the memory card file system.

Number of Open Files

The total number of open files at one time is one.

Maximum Data Per Message

The maximum amount of data that can be read or written to the memory card at one time with the application is 450 bytes. This amount is the maximum because we are using CIP Generic Messages. The application code does not permit attempts to read or write more than this amount at one time. If the code is modified to access more than 450 bytes, the data cannot be read or written to the memory card. The application code must perform and handle multiple calls to the file system when you access more than 450 bytes of data.

Removal and Insertion of Memory Card

The application does not support the removal or insertion (RIUP) of the memory card when a memory card command is active.

Minimum File Size/Resolution

The minimum file size that is supported is 450 bytes. If you create a file and write 4 bytes to the file, the application can write 450 bytes. The same concept applies for reads and writes. If you read or write 500 bytes, 900 bytes are read or written.

Memory Resources and Performance

The subroutines described in this document depend on messaging to perform the required operations to the memory card. These operations use the same memory resources that are used to perform other messaging. The performances of the memory card operations are affected by:

- the communications System Overhead Time Slice and amount of other messaging or HMI communications that are occurring in the controller. You can configure the System Overhead Time Slice from the Controller Properties dialog.
- the task structure of the program. Motion, periodic tasks, or event tasks slow performance when accessing the memory card.

See [Performance Data on page 49](#) for more information.

Controllers Supported

Only Logix5000™ controllers support read/write access to the memory card with this feature.

Limited Life

Due to the limited life of any memory card, write to the card only when necessary. Do not write to the memory card every program scan. Write to the card only at predetermined intervals, such as a shift change.

Read or Write Portions of a File

The application code does not support reading or writing to portions of an existing file. Only the entire file can be read or written. If a part of the file must be modified, read the entire file from the memory card, make any necessary changes, and then write the file back to the card.

Number of Files that Can Be Created

Currently, the application only supports the creation of files on the root directory. Because the FAT16 format is used on the memory card, you are limited to 250 files. This number is a Microsoft product limitation.

Memory Card Commands

Introduction

This chapter describes four of the memory card commands, the ControlFlash Data Working Tag, and how to determine the number of bytes to read or write.

Topic	Page
CF_Data_Working Tag	16
Create Command	17
Write Command	18
Read Command	19
Delete Command	20
Determine the Number of Bytes to Read or Write	21

The file system can perform various operations to read or write data to the memory card. There are four basic commands that you can access:

- Create command
- Write command
- Read command
- Delete command

There is an extra command, verification if a memory card is present, but do not access this command, it runs as a background check.

CF_Data_Working Tag

Before explaining the four commands, you must understand the layout of a key tag in the application. The CF_Data_Working tag, a user-defined data type tag (UDT), is at the program scope.

The following table is an example of a user-defined data type tag or UDT using the CF_Data_Working tag.

Scope: MainProgram Shgw... Show All				
Name	Style	Data Type	Description	
CF_Data_Working		CF_File_Structure		
CF_Data_Working.File_Name		String140	This is the file name. Remember t...	
CF_Data_Working.String_Length	Decimal	INT		
CF_Data_Working.File_Handle	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Bytes_to_RW	Decimal	DINT	Enter the number of bytes you wa...	
CF_Data_Working.Actual_Bytes_to_RW	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Chunks_to_RW	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Point_Method	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Pointer_Offset	Decimal	DINT[2]	Do not manipulate this value	
CF_Data_Working.Point_Move_Length	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Point_Last_Position	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Commands	Decimal	DINT	Do not manipulate this value	
CF_Data_Working.Spare	Decimal	DINT		
CF_Data_Working.Data	Decimal	SINT[4950]	This is the data that is written/read	

When a command is executed, the routines use this tag to obtain all information to perform the command, like file name, data, or number of bytes to read or write. The following table is a layout of the UDT in the file structure area.

Name	Data Type	Style	Description
File_Name	String140		This is the file name. Remember to ac
String_Length	INT	Decimal	
File_Handle	DINT	Decimal	Do not manipulate this value
Bytes_to_RW	DINT	Decimal	Enter the number of bytes you want ot
Actual_Bytes_to_RW	DINT	Decimal	Do not manipulate this value
Chunks_to_RW	DINT	Decimal	Do not manipulate this value
Point_Method	DINT	Decimal	Do not manipulate this value
Pointer_Offset	DINT[2]	Decimal	Do not manipulate this value
Point_Move_Length	DINT	Decimal	Do not manipulate this value
Point_Last_Position	DINT	Decimal	Do not manipulate this value
Commands	DINT	Decimal	Do not manipulate this value
Spare	DINT	Decimal	
Data	SINT[4950]	Decimal	This is the data that is written/read

The UDT tag type is called CF_File_Structure. This tag type is important if you create a Recipe_Manager. The only three tags to manipulate in the UDT are:

- File_Name - The name of the file that is on the memory card. This name can be up to 140 characters in length. The file name must include a file extension. The.xxx is included in the 140 characters. For example, testfile1.dat would be an 11-character file name. The format of the filename follows standard Microsoft file-naming conventions.
- Bytes_to_RW - The number of bytes that you want to read or write to a file.
- Data - The SINT array is where you place the data that you want to write or where data read is put. The Baseline application supports up to 4500 bytes of data, but can easily be expanded.

See [Expand the Code to Support More Than 4500 Bytes of Data on page 55](#), for more information.

Do not manipulate the other tags. They are used by the routines as they execute; manipulation of tags could cause unexpected changes to the file currently opened.

Create Command

Use the Create command when a file does not exist on the memory card and you want to create one. A file must exist on the memory card before you can use the other three commands.

The following operations occur when you use the Create command:

- Sets the file attribute parameters - You do not need to manipulate the file attribute parameters.
- Creates the file - When a Create command is performed, the file is created and then opened.
- Closes the file.
- Error handling is performed, if needed.

Set Up the Create Command

To configure the Create command, define the filename including the file extension, for example, testfile1.dat. To execute the Create command, set the execute_CF_Create_File bit. The application code resets this bit.

IMPORTANT Use the Write command to write data to the memory card. A file must exist on the memory card before executing this command.

Write Command

The following 11 operations occur when you use write commands:

- Data parsing - Only 450 bytes can be written at one time to the file. Therefore, writing to the file is broken down into 450-byte chunks. This command moves the data into the data_to_write array from the data array in the CF_Data_Working UDT that puts in the place holders for the checksums. It also verifies that all arrays used are large enough to handle the data.
- Checksum calculations - This command calculates the checksum for each 450-byte chunk of data and places it into the place holders that are created in Data parsing.
- The file is open.
- The file pointer is set.
- The data is written.
- The file is closed.
- The file is opened.
- The file is read.

Only 450 bytes can be read at one time from the file. Therefore, reading from the file is broken down into 450-byte chunks. This process moves the data into the data_read array.

- The checksums are verified and the data in data_to_write array and data_read array are compared to help ensure that they are equal.

TIP If the checksums are not calculated, compared, and parsed, the file is closed and the write sequence is restarted from the beginning.

The application attempts to write a file four times. After the fourth failure, a checksum error is set.

- The file is closed.
- Error handling is performed, if needed.

Set Up the Write Command

To configure the write command, you must set the `execute_CF_Write_Data` bit. The application code handles the reset of this bit.

1. Define a file name including a file extension, for example, `testfile1.dat`.
2. Enter the number of bytes you want to write to the file.
3. Move the data to be written in the `CF_Data_Working` UDT into the data array.

We recommended that you clear the data array before moving the data into the array. This process clears out any data from previous read or write commands.

Read Command

IMPORTANT The read command is used when you want to read data from the memory card. A file must exist on the memory card before executing this command.

The following six operations occur when you use read commands:

- The file is open.
- The file pointer is set.
- The file is read.

Only 450 bytes can be read at one time from the file. Therefore, reading from the file is broken down into 450-byte chunks. This process moves the data into the `data_read` array.

- The checksums are calculated and compared to checksums that are in the file. The data is then parsed into the Data array in the `CF_Data_Working` UDT.

TIP If the checksums are not calculated, compared, and parsed, the file closes and the read sequence is restarted from the beginning.

The application attempts to read the file four times. After the fourth failure, a checksum error is set.

- The file is closed.
- Error handling is performed, if needed.

Set Up the Read Command

To configure the read command, you must set the `execute_CF_Read_Data` bit. The application code handles the reset of this bit.

1. Define a file name including a file extension, for example, `testfile1.dat`.
2. Enter the number of bytes you want to read from the file.
3. Move the data from the data array in the `CF_Data_Working` UDT when the read is completed.

Only 450 bytes can be read at one time from the file. Therefore, reading from the file is broken down into 450-byte chunks. This process moves the data into the `data_read` array.

We recommended that you clear the data array before moving the data into the array. This process clears out any data from previous read or write commands.

See [Chapter 4, Example Application](#) for complete instructions.

Delete Command

Use the Delete command when a file must be deleted from the memory card. A file must exist on the memory card before you can use the Delete command.

The following two operations occur when you use delete commands:

- Delete the file.
- Error Handling is performed, if needed.

Set Up the Delete Command

To configure the Delete command:

- define a filename including a file extension, for example, `testfile1.dat`.
- set the bit `execute_CF_Delete_File`. The application code handles the reset of this bit.

See [Chapter 4, Example Application](#) for complete instructions.

Determine the Number of Bytes to Read or Write

Tags of any data type, including arrays, user-defined types (UDT), and arrays of UDT tags can be read and written to the memory card.

When reading and writing data to the memory card, you must know how many bytes are actually being read or written. [Table 3](#) indicates the type and size of the atomic data types. The minimum memory allocation for a tag is 4 bytes. When you create a tag that stores data that requires less than four bytes, the controller allocates four bytes, but the data only fills the part that it needs.

Table 4 - Data Types

Data Type	Bits						
	31	16	15	8	7	1	0
BOOL	Not used						0 or 1
SINT	Not used				-128¼+127		
INT	Not used		-32,768¼32767				
DINT	-2,147,483,648¼+2,147,483,647						
REAL	-3.40282347E ³⁸ ¼-1.17549435E ⁻³⁸ (negative values) 0 1.17549435E ⁻³⁸ ¼3.40282347E ³⁸ (positive values)						

For example, if one DINT type tag value is being written to the memory card, 4 bytes is the amount of data that is written. If an array of DINT tags is being written, the number of bytes being written is four times the size of the array.

For example, the tag MYDINTS[100] is 4 times 100 elements, or 400 bytes of data. Similarly, if a UDT is used, one instance of that UDT is the sum of the sizes of all data types within the structure.

TIP When reading or writing to the memory card, it is best to use UDT tags. Use of these tags let you mix different data types together and lets you know the number of bytes that must be transferred to and from the card.

The following is a UDT that is the recipe to make waffles.

Name:Recipe_Data

Description:Recipe for making waffles

Members:Data Type Size: 72 byte(s)

	Name	Data Type	Style	Description
	Number_of_Eggs	DINT	Decimal	
	Cups_of_Oil	REAL	Float	
	Cups_of_Water	REAL	Float	
	Cups_of_Milk	REAL	Float	
	Cups_of_Mix	REAL	Float	
	Add_Blue_Berries	BOOL	Decimal	
	Add_Choc_Chips	BOOL	Decimal	
	Mix_Time	TIMER		
	Cook_Time	TIMER		
	Cook_Temperature	TIMER		
	Cool_Time	TIMER		

As you can see, the total size of the recipe is 72 bytes. This size means that you would want to read or write 72 bytes of data to the memory card. This data would be entered into the Bytes_to_RW element of the CF_File_Structure UDT.

Example Application

Introduction

This chapter describes how to use command in an example application. The first example uses a ControlLogix® 5560 and a CompactFlash card, and the second uses a ControlLogix 5570 and an SD card.

Topic	Page
Requirements	23
Configure Message Instructions	24
Create a File on the Memory Card	25
Write Data to the Memory Card	28
Read Data from the Memory Card	30
Delete a File on the Memory Card	32
Build an Example	33

Requirements

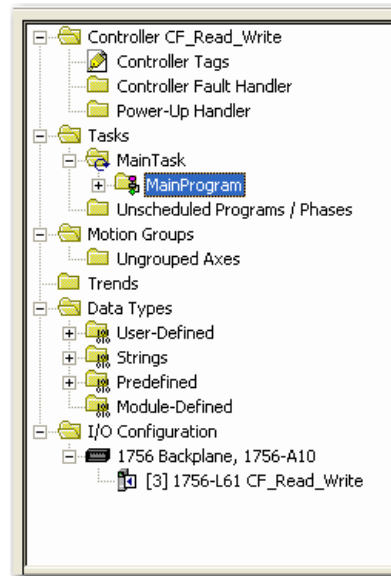
For this example, you must have the RSLogix 5000® software, version 16.000 or later, installed and open. Also, a memory card must be installed in the controller. The application detects if a card is installed.

Configure Message Instructions

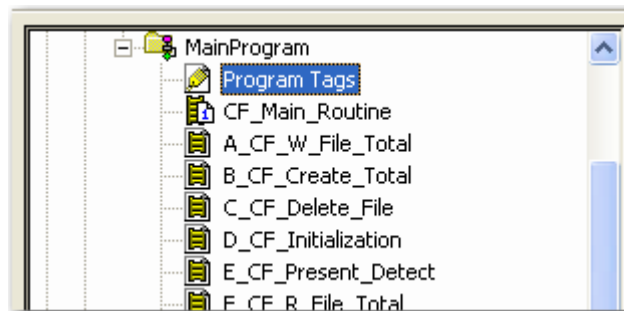
Most of this configuration has already been performed for you in the ACD file. The only configuration that is needed is to set the controller slot number.

Set the Controller Slot Number

1. Open the project CF_Read_Write_VXX_XX.ACD
2. Open the MainPrograms tags.

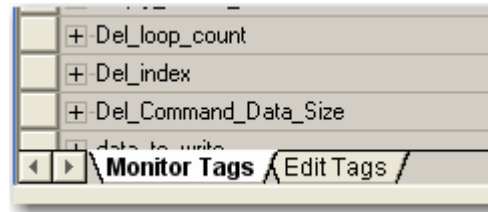


3. Expand the MainPrograms tags.

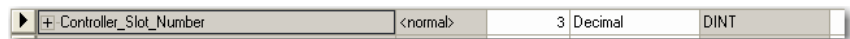


4. Double-click Program Tags.

Make sure that the Monitor Tags tab is selected in the tag editor.



5. Sort the tags by Name.
6. Find the tag Controller_Slot_Number.



7. Enter the slot number where the controller is located in the value column.

When the application enters Run mode, the D_CF_Intialization routine executes. The paths for all message instructions are modified to contain the slot number you just entered. You can add a new rung zero to the D_CF_Intialization routine to move the slot number into the tag Controller_Slot_Number.

8. Download the project to the controller and go online.
9. Put the controller into Run mode.

Create a File on the Memory Card

Before you can perform any read or write commands to a file, you must create the file on the memory card first.

1. Open the MainPrograms tags.

Make sure that the Monitor Tags tab is selected in the tag editor.

2. Sort the tags by Name.
3. Find the CF_Data_Working tag.

This tag is a user-defined data type (UDT).

4. To expand the CF_Data_Working tag, Click the + next to the tag.

The Filename element of the UDT CF_Data_Working must be modified.


Filename, as it implies, is the name of the file you want to perform an operation on, in this case create. The file name can be up to 140 characters in length. You must include a file extension.

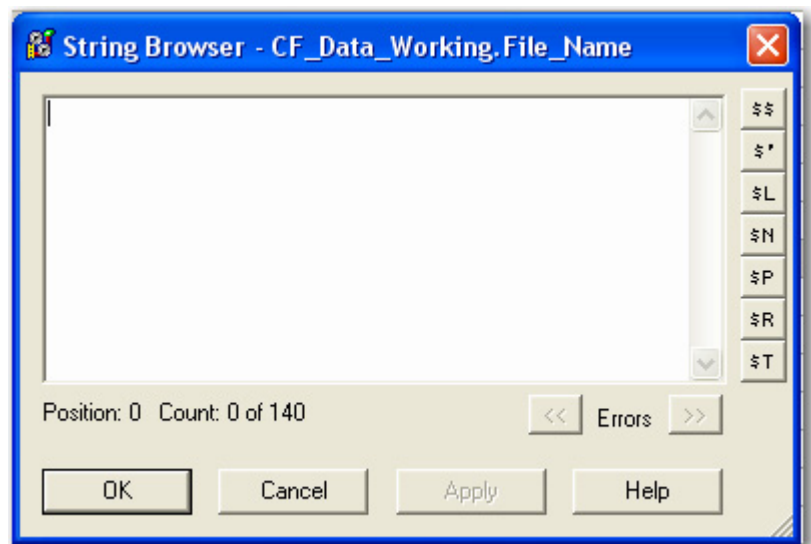
5. Click in the value field.



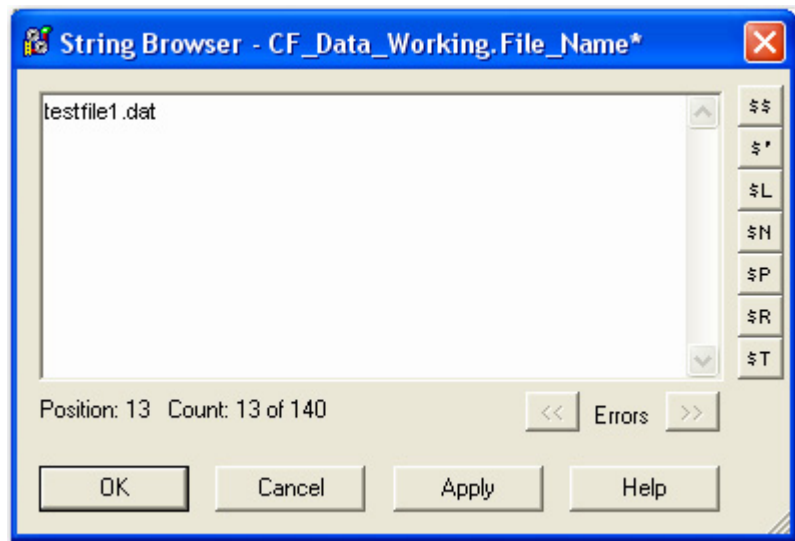
The Browse button appears.



6. Click Browse  and the string browser window displays.



7. Enter testfile1.dat and click OK.



8. Switch the Scope to CF_Read_Write in the tag monitor to view the Controller Scoped tags.



9. Sort the tags by name.
10. Find the execute_CF_Create_File tag.
11. Enter a 1 in the value field and press Enter.

The tag returns to a 0. This value indicates that the application has attempted to create your file.

12. Find the CF_Create_Seq_Status tag.
13. Click + to expand the tag.

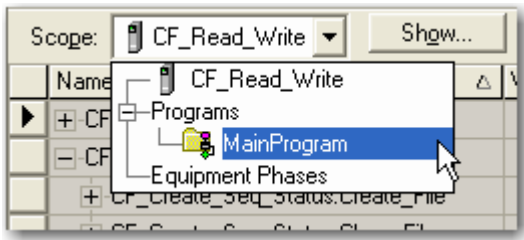
This action is a user-defined data type (UDT). As the tag name indicates, this UDT gives you status on the create file sequence. There is an element of the UDT called Overall_Status, which is equal to 1. This value means that the file was created successfully.

If another value is present, refer to [AppendixC, Troubleshoot the Memory Card File System](#).

Write Data to the Memory Card

Configure how the program is going to write the data to the memory card.

1. Switch to the MainPrograms tags.



Make sure that the Monitor Tags tab is selected in the tag editor.

2. Sort the tags by Name.
3. Find the CF_Data_Working tag.

This tag is a user-defined data type (UDT).

4. Click + to expand the tag.

[-] CF_Data_Working	{...}
+ CF_Data_Working.File_Name	'testfile1.dat'
+ CF_Data_Working.String_Length	0
+ CF_Data_Working.File_Handle	0
+ CF_Data_Working.Bytes_to_RW	0
+ CF_Data_Working.Actual_Bytes_to_RW	0
+ CF_Data_Working.Chunks_to_RW	0
+ CF_Data_Working.Point_Method	0
+ CF_Data_Working.Pointer_Offset	{...}
+ CF_Data_Working.Point_Move_Length	0
+ CF_Data_Working.Point_Last_Position	0
+ CF_Data_Working.Commands	0
+ CF_Data_Working.Spare	0
+ CF_Data_Working.Data	{...}

Leave the filename the same as the previous section, testfile1.dat. There is an element in the UDT called CF_Data_Working.Bytes_to_RW. This element tells the application how many bytes in this case are written to the file.

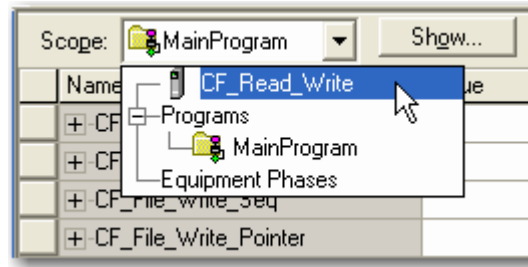
5. Find CF_Data_Working.Bytes_to_RW and enter a value of 450.

This entry means that 450 bytes are written to the file testfile1.dat.

6. Expand the data element in the UDT (CF_Data_Working.Data).

This array holds the data to be written to the file.

7. Enter a value of 100 in the first element of the Data array.
8. To view the Controller Scoped tags, switch the Scope to CF_Read_Write in the tag monitor.



9. Sort the tags by name.
10. Find the execute_CF_Write_Data tag.
11. Enter a 1 in the value field and press Enter.

The tag returns to a 0. This value indicates that the application has attempted to write to your file.

12. Find the CF_Write_Seq_Status tag and click + to expand the tag.

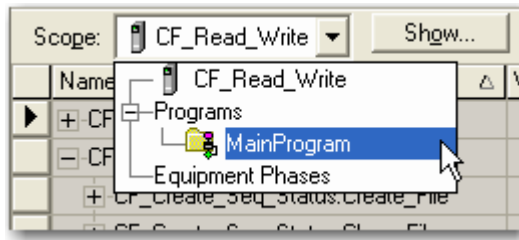
This tag is a user-defined data type (UDT). As the tag name indicates, you receive status on the write file sequence. There is an element of the UDT called Overall_Status. It is equal to 1. This value means that the file was written successfully.

If another value is present, refer to [AppendixC, Troubleshoot the Memory Card File System](#).

Read Data from the Memory Card

Determine how the program is going to read the data from the memory card.

1. Switch to the MainPrograms tags.



Make sure that the Monitor Tags tab is selected in the tag editor.

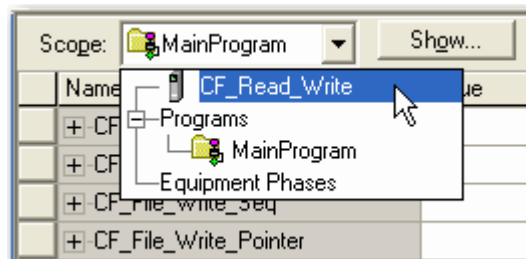
2. Sort the tags by Name.
3. Find the CF_Data_Working tag.

This tag is a user-defined data type (UDT).

4. Click + to expand the tag.
5. Use the file name from the previous section, testfile1.dat.

The UDT element Bytes_to_RW remains set at 450. The UDT element that is called Bytes_to_RW tells the application how many bytes in this case are read from the file.

6. Enter a value of 127 in the first element of the data array (CF_Data_Working.Data).
7. Switch to view Controller Scoped tags in the tag monitor.



8. Sort the tags by name.
9. Find the execute_CF_Read_Data tag.

10. Enter a 1 in the value field and press Enter.

The tag returns to a 0. This value indicates that the application has attempted to read from your file.

11. Find the CF_Read_Seq_Status tag and click + to expand the tag.

This tag is a user-defined data type (UDT). As the tag name indicates, you receive status on the read file sequence. There is an element of the UDT called Overall_Status. It is equal to 1. This value means that the file was created successfully.

If another value is present, refer to [AppendixC, Troubleshoot the Memory Card File System](#).

12. Switch to view MainProgram Scoped tags in the tag monitor.
13. Find the CF_Data_Working tag.

This tag is a user-defined data type (UDT).

14. Click + to expand the tag.
15. Expand the SINT array Data in the UDT.

Earlier in this section, you entered a value of 127. The value is now 100, which is what you wrote to the file testfile1.dat in the last section and was read back.

Delete a File on the Memory Card

Determine how the program is going to delete a file from the memory card.

1. Open the MainPrograms tags.

Make sure that the Monitor Tags tab is selected in the tag editor.

2. Sort the tags by Name.
3. Find the CF_Data_Working tag.

This tag is a user-defined data type (UDT). Expand the UDT by clicking in the + next to the tag. There are no elements of the UDT CF_Data_Working that must be modified. The file name testfile1.dat is already entered. You want to delete this file.

4. Switch to view Controller Scoped tags in the tag monitor.
5. Sort the tags by name.
6. Find the execute_CF_Delete_File tag.
7. Enter a 1 in the value field and press Enter.

The tag returns to a 0. This value indicates that the application has attempted to delete your file.

8. Find the CF_Delete_Seq_Status tag and click + to expand the tag.

This tag is a user-defined data type (UDT). As the tag name indicates, you receive status on the file sequence. There is an element of the UDT called Status. It is equal to 1. This value means that the file was deleted successfully.

If another value is present, refer to [AppendixC, Troubleshoot the Memory Card File System](#).

Build an Example

Complete the following steps to build an example application with the waffle recipe from the last chapter.

1. Open the file CF_Read_Write_VXX_XX_Example.ACD.

This code handles four recipes.

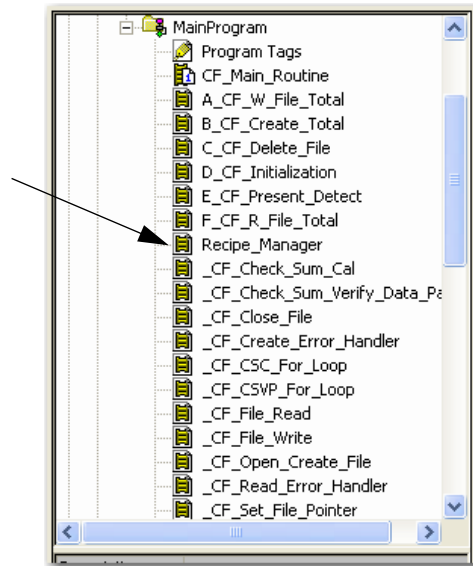
In the controller organizer under the MainProgram, you see a routine called Recipe_Manager. This routine shows a basic recipe manager and how to interface into the Baseline memory card application.

Rung zero is used to make sure that you do not try to access two recipes simultaneously or change a memory card command when one is executing. Enter a value of 1, 2, 3, or 4 in Recipe_Number and a value of 1 to 4, depending on the memory card command you want into Recipe_Command.

1 = create
2 = write
3 = read
4 = delete

Four rungs remain and each rung is for another recipe. In this example, you are only going to examine one of them.

2. Open the Recipe_Manager routine.



3. See the next rung (rung 1).

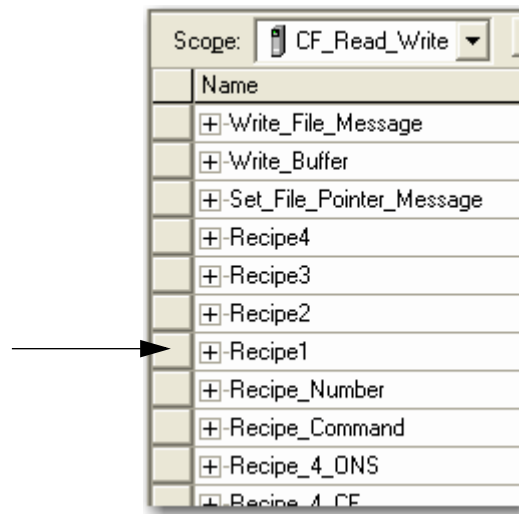
The first conditions determine that this recipe and a command have been selected. After that determination, there are five branches.

Branch	Description
1	Handles the command to write data to the memory card.
2	Handles creation of the file on the memory card.
3	Handles the reading of data from the memory card and placement into the Recipe.
4	Handles deletion of the file on the memory card.
5	Handles clearing the Recipe_Manager for the next recipe and command.

Set Up Recipe Data

The next item to do is to set the application to perform commands to the memory card. First, you must configure recipe data to store to the card.

1. Open the Controller Tags from the controller organizer.
2. Click Monitor Tags.
3. Scroll down until you find the Recipe1 tag.



This tag is the UDT (Recipe Data) we discussed in the previous chapter.

- Expand the tag and enter some data.

+ Recipe1.Number_of_Eggs	10	Decimal
- Recipe1.Cups_of_Oil	2.0	Float
- Recipe1.Cups_of_Water	2.0	Float
- Recipe1.Cups_of_Milk	1.0	Float
- Recipe1.Cups_of_Mix	12.0	Float
- Recipe1.Add_Blue_Berries	1	Decimal
- Recipe1.Add_Choc_Chips	1	Decimal

Earlier we discussed a program scope tag called CF_Working_Data. When a command is executed, the routines use this tag to get all information, such as file name, data, number of bytes to read or write.

In the [Memory Card Commands](#) chapter, we manipulated data directly in this tag, but since we are now interested in multiple files, we will not. New tags are created that have this type for each recipe. The Recipe_Manager code handles the copying of this data.

- Find the Recipe_1_CF tag in the controller organizer.
- Expand the tag.

The first element to manipulate is the File_Name.

- Enter recipe1.dat.

Remember to use a file extension.

- Define how many bytes you want to read/write to the file.

From the previous section, a waffle recipe was 72 bytes in size.

- Enter 72 in the Bytes_to_RW tag.

The last piece of data to manipulate is the Data. Copy the data from the waffle Recipe1 into this array. This process is completed in the Recipe_Manager code.

TIP Remember to set the tag controller_slot_number and to make sure that a memory card is installed in the controller.

10. Save the project and download it to the controller.
11. Go online with the controller.

You are now ready to read and write your recipe data to the memory card.

12. Open the Controller tags and go to the Monitor tab.
13. Look for the Recipe_Command and Recipe_Number tags.

Use the first recipe as an example.

Create the File

Tell the Recipe_Manager routine to create a file.

1. Enter a value of 1 in the Recipe_Number tag.

This value tells the Recipe_Manager to work with Recipe1.

2. Enter a value of 1 in the Recipe_Command tag.

This value tells the Recipe Manager to create the file.

Once the command has completed, the tags Recipe_Number and Recipe_Command are set to zero.

Write the Recipe

Write the recipe data to the memory card. You entered data into Recipe1 before you download to the controller.

1. Enter a value of 1 in the Recipe_Number tag.

This value tells the Recipe_Manager routine to work with Recipe1.

2. Enter a value of 2 in the Recipe_Command tag.

This value tells the Recipe_Manager to write to the file. Once the command has completed, the tags Recipe_Number and Recipe_Command are set to zero.

Read the Recipe

Before you read the recipe from the memory card, you must clear out the values you entered earlier.

1. Go to the Recipe1tag and clear out the values you entered earlier.
2. Enter a value of 1 in the Recipe_Number tag.

This value tells the Recipe_Manager to work with Recipe1.

3. Enter a value of 3 in the Recipe_Command tag.

This value tells the Recipe_Manager to read from the file.

Once the command has completed, the tags Recipe_Number and Recipe_Command are set to zero.

Now look back in the tag Recipe1 the values have been read from the memory card and the values have been restored.

Delete the Recipe

Tell the Recipe_Manager routine to delete the file.

1. Enter a value of 1 in the Recipe_Number tag.

This value tells the Recipe_Manager to work with Recipe1.

2. Enter a value of 4 in the Recipe_Command tag.

This value tells the Recipe_Manager to delete a file. Once the command has completed, the Recipe_Number and Recipe_Command tags are set to zero.

Format a Memory Card

Introduction

This appendix describes how to format a memory card.

Topic	Page
Requirements	39
Format with a Personal Computer and a Memory Card Reader	39
Format a CompactFlash Card with RSLogix 5000 Software	41
Format an SD Card with the Logix Designer Application	45

Requirements

To use the information that is described in this document, it is necessary to have the memory card that is formatted to an FAT16 file system. Allen-Bradley® memory cards (1784-CF64) are pre-formatted to an FAT16 file system and are ready to use. You can format the memory card in the following ways:

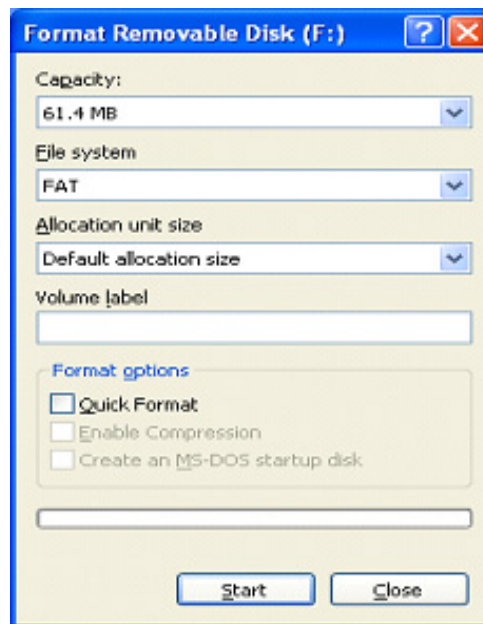
- by using a personal computer with a memory card reader
- RSLogix 5000 software, version 20 or earlier
- Logix Designer application, version 21 or later

Format with a Personal Computer and a Memory Card Reader

You can use a Windows-based operating system on a personal computer with a memory card reader to format the card. The following steps illustrate how to do this using Windows XP Professional. If another version of Windows is being used, the steps are similar. You can find detailed steps in the Windows online help or the instruction manual.

1. Insert the memory card into the memory card reader.
2. Go to My Computer with the icon on the desktop.
3. Right-click the drive that corresponds to the memory card reader and select Format.

The following screen appears.



4. Choose FAT from the File system pull-down menu.
5. Enter a volume label, if desired.
6. Click Start.

These actions format the card and, once complete, the card is ready for use.

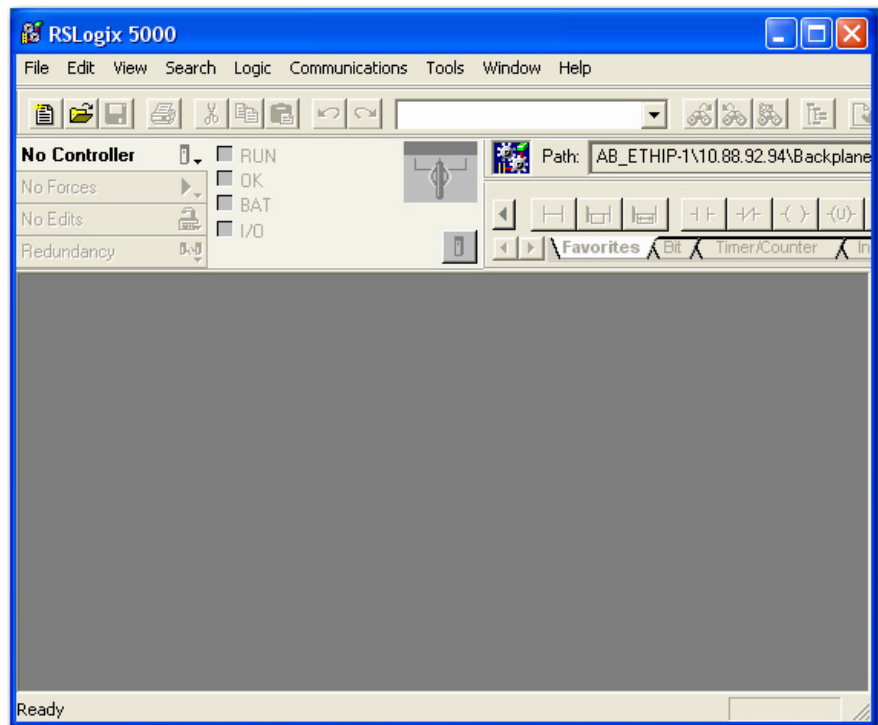
The second method formats the card only if it is unformatted or incorrectly formatted. In both cases, if the card is formatted, all data on the card is lost.

Format a CompactFlash Card with RSLogix 5000 Software

1. Open RSLogix 5000® software, click Start > Programs > Rockwell Software > RSLogix 5000 Enterprise Series > RSLogix 5000.

The following screen appears.

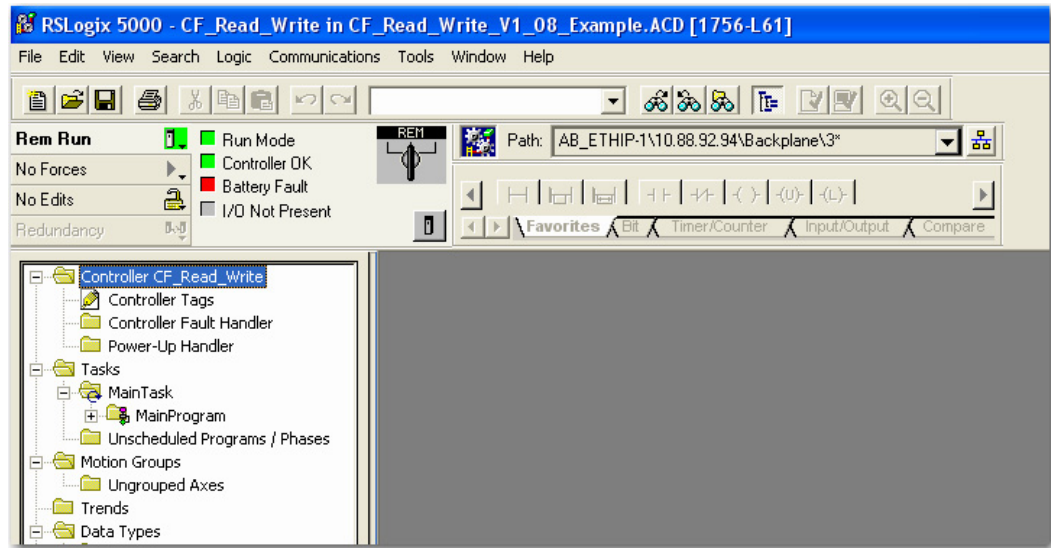
2. Select File > Open.



3. Navigate to the location of CF_Read_Write_V3_2 or any other file that is compatible with RSLogix 5000 software, version 16 or later.

4. Open the file.

The following window appears. You could see a slightly different image if you are not using the provided example, but the functionality is the same.

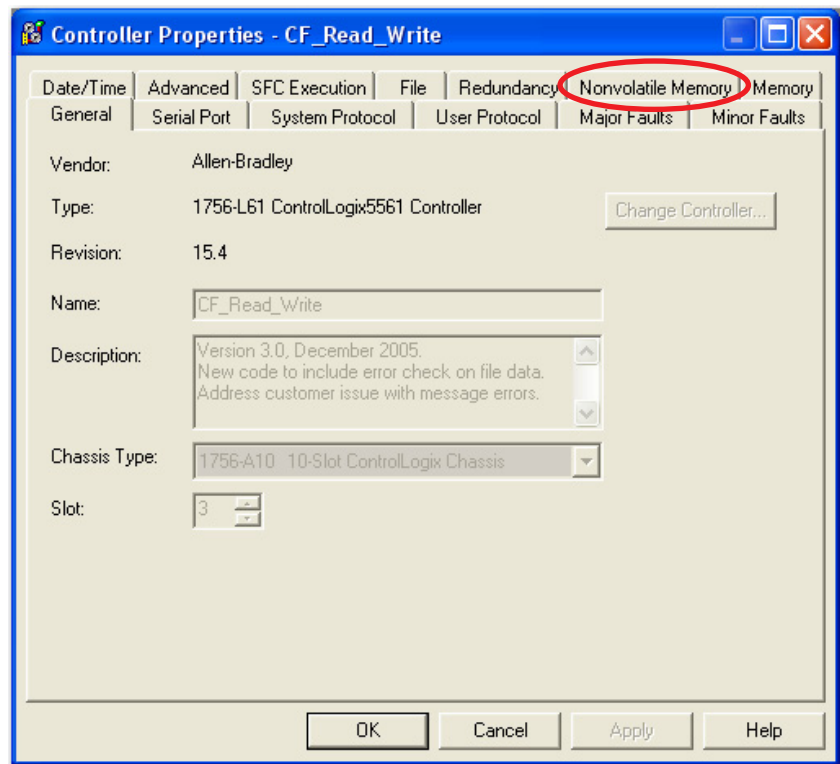


Be sure that the controller is Online and in Program mode.

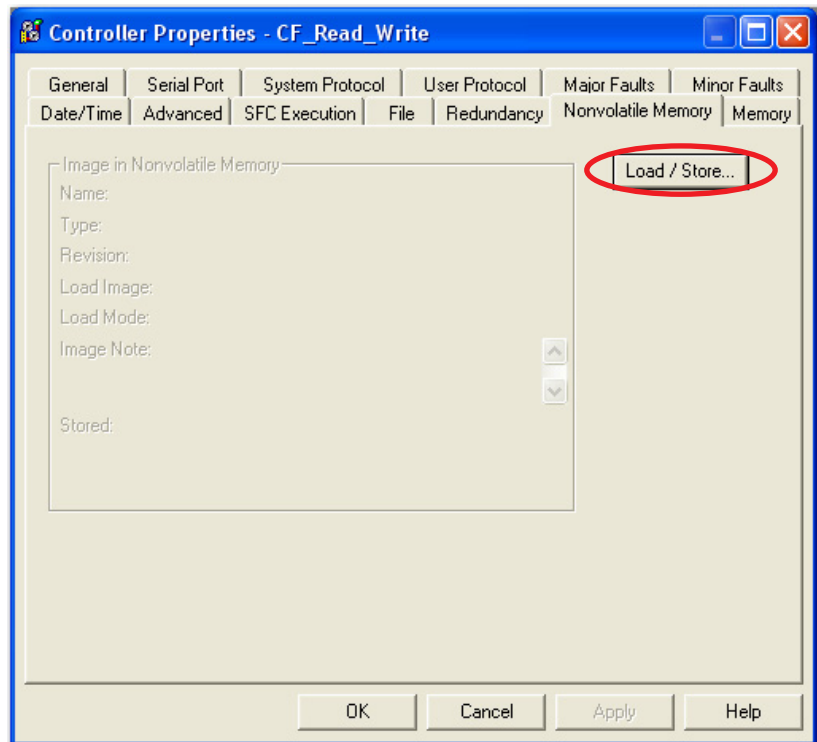
5. Right-click the Controller folder in the Controller Organizer and select Properties.

The Controller Properties dialog box appears.

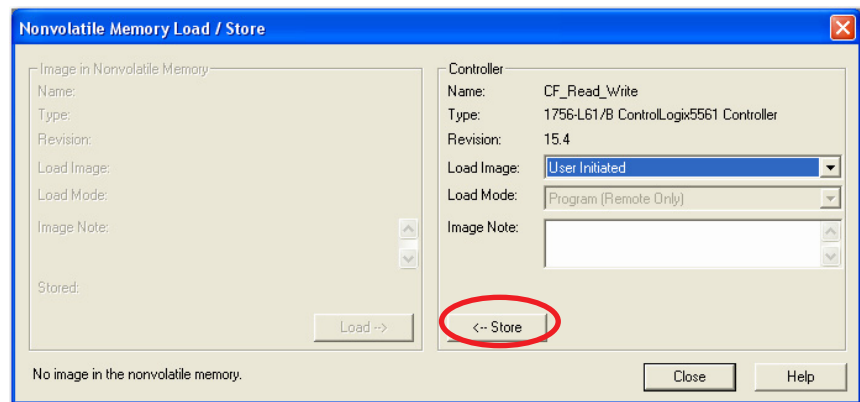
6. Click the Nonvolatile Memory tab.



7. Click Load/Store on the Nonvolatile Memory tab.

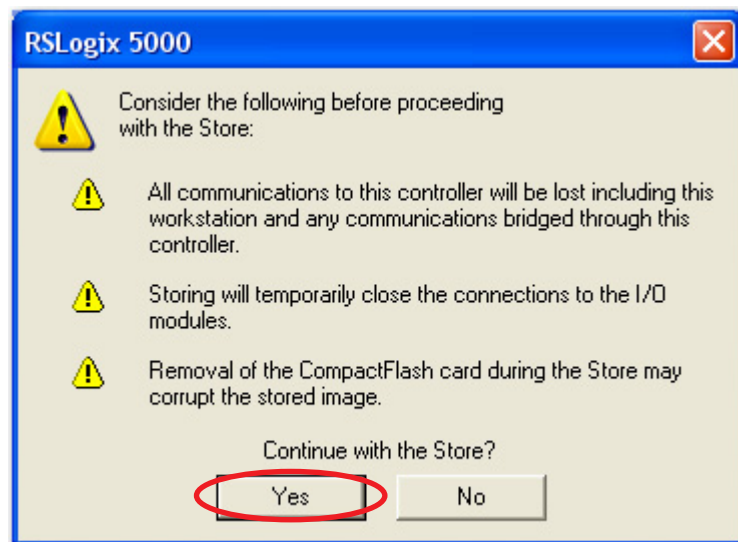


8. Click Store.

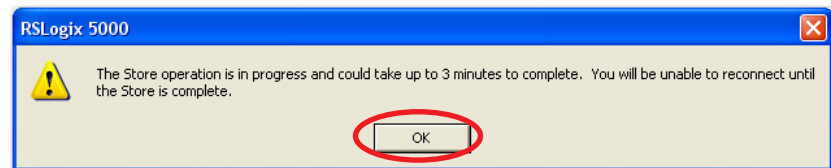


The following message box appears.

9. Click Yes.



10. Click OK in any dialog boxes that can appear.



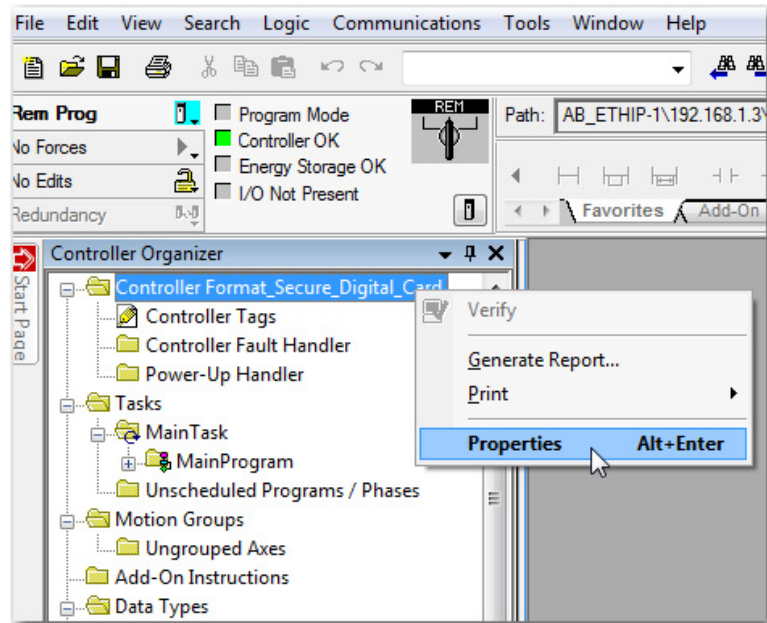
The memory card formats to an FAT16 file system and the current project image file is stored on the memory card. The card is now ready for use.

Format an SD Card with the Logix Designer Application

1. Open Logix Designer® click Start > Programs > Rockwell Software > Studio 5000.
2. Select File > New.
3. Name the file. In this example, it is named Controller_Format_Secure_Digital_Card.

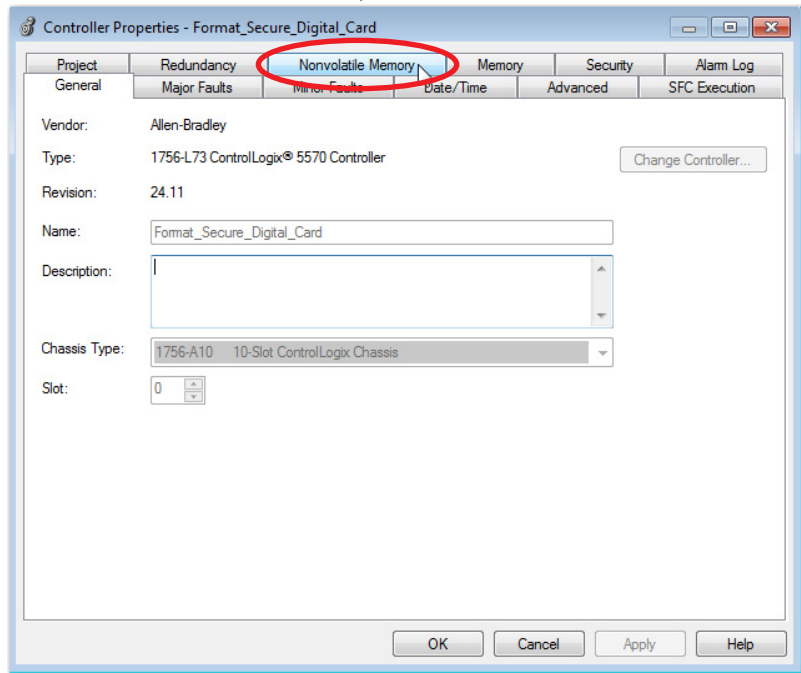
Be sure that the controller is Online and in Program mode.

4. Right-click and select properties.

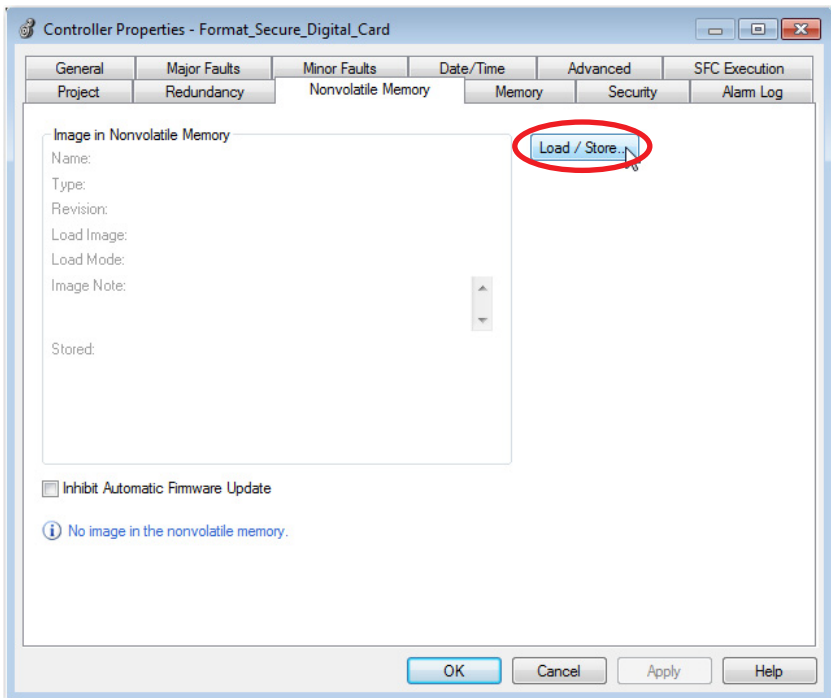


The Control Properties dialog box appears.

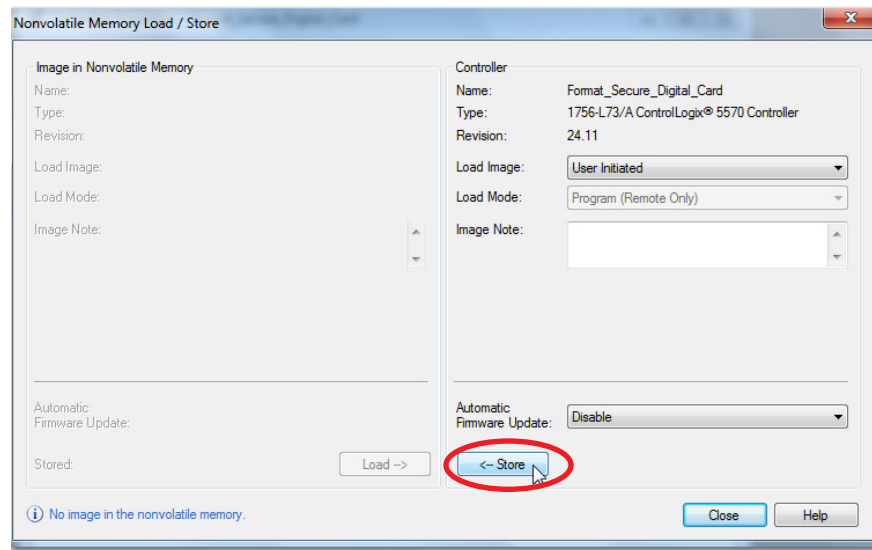
5. Click the Nonvolatile Memory tab.



6. Click Load/Store on the Nonvolatile Memory tab.

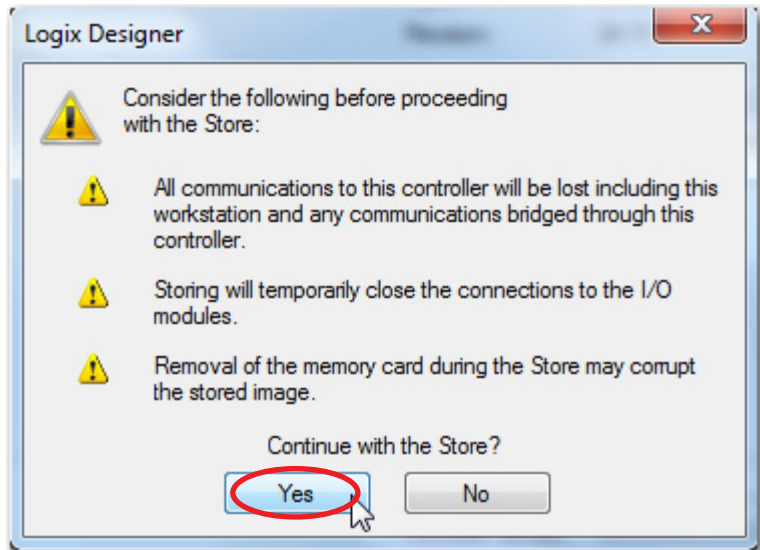


7. Click Store.

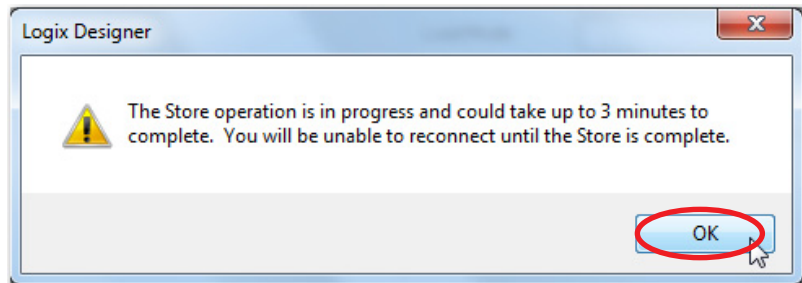


The following message box appears.

8. Click Yes.



9. Click OK in any dialogs that can appear.



Performance Data

Introduction

This appendix describes performance data of the memory card. Performance with the use of third-party memory cards vary.

Topic	Page
Performance Data Tables	50
Baseline Performance When Accessing 450 Bytes	50
Baseline Performance When Accessing 4500 Bytes	50

TIP The application that is described in this document depends on messaging to perform operations. These are operations such as open, read, or write, on the memory card.

Therefore, time slice, and the amount of other messaging and HMI communications occurring in the controller affect data transfer rates. These factors also affect program structures, for example, motion, periodic tasks, and event tasks.

Performance Data Tables

The following tables show the performance of the memory card in different instances. The values express the typical number of milliseconds it takes to execute a command.

Baseline Performance When Accessing 450 Bytes

This table shows the Baseline performance of the memory card. The sample application code was used to access 450 bytes. All commands were performed multiple times and the typical value is shown.

	Create Command*	Write Command		Read Command		Delete Command*
		Command Execution Time	Program Scan Time ⁽³⁾	Command Execution Time	Program Scan Time ⁽³⁾	
ControlLogix 5560⁽¹⁾	67 ms	187 ms	25 ms	52 ms	5 ms	82 ms
ControlLogix 5570⁽²⁾	19 ms	79 ms	11 ms	30 ms	2 ms	13 ms
CompactLogix 5380⁽²⁾ ControlLogix 5580⁽²⁾	16 ms	55 ms	1 ms	16 ms	0.5 ms	16 ms

(1) Tests were completed with 1784-CF64 Revision B01

(2) Tests were completed with 1784-SD1/A

(3) Represent a short term temporary increase in the program scan when the command is executed.

*Create and delete commands have minimal affect on program scan.

Baseline Performance When Accessing 4500 Bytes

This table shows the Baseline performance of the memory card. The sample application code was used to access 4500 bytes. All commands were performed multiple times and the typical value is shown.

	Create Command*	Write Command		Read Command		Delete Command*
		Command Execution Time	Program Scan Time ⁽³⁾	Command Execution Time	Program Scan Time ⁽³⁾	
ControlLogix 5560⁽¹⁾	68 ms	572 ms	45 ms	178 ms	25 ms	83 ms
ControlLogix 5570⁽²⁾	19 ms	390 ms	20 ms	125 ms	10 ms	16 ms
CompactLogix 5380⁽²⁾ ControlLogix 5580⁽²⁾	16 ms	330 ms	3 ms	33 ms	2 ms	16 ms

(1) Tests were completed with 1784-CF64 Revision B01

(2) Tests were completed with 1784-SD1/A

(3) Represent a short term temporary increase in the program scan when the command is executed.

TIP Significant amounts of HMI or messaging traffic can affect the times that are shown in the preceding table.

*Create and delete commands have minimal affect on program scan.

Troubleshoot the Memory Card File System

Introduction

This appendix describes possible troubleshooting for specific topics that are encountered while using the memory card file system.

Topic	Page
Status User-defined Data Type	51
Tag Layout	52
Message Error Codes	53

Status User-defined Data Type

For each command that the application supports there is a status user-defined data type created. These data types are found in the Controller Scoped tags.

Table 5 - Controller Scoped Tags

Command	Controller Scoped Tag
Create a file	CF_Create_Seq_Status
Write to a file	CF_Write_Seq_Status
Read from a file	CF_Read_Seq_Status
Delete a file	CF_Delete_Seq_Status

The formats are generally the same, however, it depends on the number of steps in each command. This example describes the CF_Write_Seq_Status tag.

[-] CF_Write_Seq_Status	CF_Write_Seq_Status		
▶ [+]	CF_Write_Seq_Status.Parse_Data	Sequence_Status	
	[+] CF_Write_Seq_Status.File_Open	Sequence_Status	
	[+] CF_Write_Seq_Status.Set_Point	Sequence_Status	
	[+] CF_Write_Seq_Status.Write_Data	Sequence_Status	
	[+] CF_Write_Seq_Status.Close_File	Sequence_Status	
	[+] CF_Write_Seq_Status.Read_File	Sequence_Status	
	[+] CF_Write_Seq_Status.Verify_Check_Sum	Sequence_Status	
	[+] CF_Write_Seq_Status.Overall_Status	DINT	Decimal

Tag Layout

In the write sequence, there are seven major steps and one overall status. The CF_Write_Seq_Status.Parse_Data.Status tag has been expanded to show its format. After a command has been executed, you will want to check the overall status.

[-] CF_Write_Seq_Status	CF_Write_Seq_Status	
[-] CF_Write_Seq_Status.Parse_Data	Sequence_Status	
▶ [+ CF_Write_Seq_Status.Parse_Data.Status	DINT	Decimal
[+ CF_Write_Seq_Status.Parse_Data.Mes...	DINT	Hex
[+ CF_Write_Seq_Status.Parse_Data.Mes...	DINT	Hex
[+ CF_Write_Seq_Status.Parse_Data.Seq...	DINT	Decimal

The following table lists the codes that appear in the overall status.

Table 6 - Overall Status Codes

Code	Description
0	The command has not been run or is running.
1	The command executed without errors.
2	The command ran with errors and you must expand each individual sequence and check its Status element.
3	A checksum error has been found. This error only occurs for the read and write commands. The error means that a read or write was attempted four times and failed the checksum verification.
8	No memory card was present when a command was executed. The command is not attempted.
Codes that appear in the individual sequence status when a 2 appears in the overall status:	
4	A file size mismatch occurred. If it is a read command, then the tag data_read is too small. If it is a write command, then data_to_write is too small. Increase the size of the array.
5	The number of Bytes_to_RW was set to zero.
6	A message instruction that is involved in this sequence errored. Examine the Message_Error and Message_Error_Extended elements.
7	No file name was entered.

Message Error Codes

The following table lists potential error codes while working with the memory card file system. All error codes are in hex.

Error / Extended	Extended Error Description
0C/0	File does not exist. Invalid handle (State conflict).
05/00	Card was removed while command was active.
02/0	Media does not have the free space to create file.
0D/0	File exists.
0C/0	File does not exist (Delete command).
10/0	Device State Conflict: the update card is not formatted.
13/0	Insufficient command data: the command packet does not contain enough data.
FF/2100	Privilege failure.

For other error codes, refer to Logix5000™ Controllers General Instructions, publication 1756-RM003.

Notes:

Expand the Code to Support More Than 4500 Bytes of Data

Introduction

This appendix describes how to expand the code to support more than 4500 bytes of data.

Topic	Page
Modify the Application to Access More Data	55
Access 10,000 Bytes Example	56
Modify the Arrays	57
Modify the Data Element	57

Modify the Application to Access More Data

The Baseline® application supports reading or writing 4500 bytes of data to the memory card. If you must access more data, modify the Baseline application.

One example is that you have three groups of data that you want to write to the memory card, one is 250 bytes, one is 3000 bytes, and one is 10,000 bytes. The first two do not warrant any changes to the Baseline application, but the last will. You must expand three of the tag array sizes.

Before writing to the memory card, calculate how large to make those three arrays. Do not increase them to 10,000 bytes. Be sure to consider that of the 450 bytes, 8 bytes are checksum information, which leaves 442 bytes for user data.

Step 1:

$$\text{Actual_Bytes_to_RW} = \text{Bytes_to_RW} + (\text{TRN}(\text{Bytes_to_RW}/442) * 8) + (\text{If the MOD}(\text{Bytes_to_RW}/442) > 0 \text{ then } 8 \text{ else } 0)$$

Step 2:

$$450 [\text{TRN}(\text{Actual_Bytes_to_RW}/450) + (\text{if the MOD}(\text{Actual_Bytes_to_RW}) > 0 \text{ then } 1 \text{ else } 0)]$$

Access 10,000 Bytes Example

For example, if you want to access 10,000 bytes follow these steps.

Step 1:

$$\text{Actual_Bytes_to_RW} = 10,000 + (\text{TRN}(10,000/442) * 8) + (\text{If the MOD}(10,000/442) > 0 \text{ then } 8 \text{ else } 0)$$

$$= 10,000 + 22(8) + 8$$

$$= 10,184$$

Step 2:

$$450[(\text{TRN}(10,184/450)) + (\text{If the MOD}(10,148/450) > 0 \text{ then } 1 \text{ else } 0)]$$

$$= 450 [22+1]$$

$$= 450(23)$$

$$= 10350 \text{ bytes}$$

You must increase the arrays to a minimum of 10,350 elements. Do not place this value in the Actual_Bytes_to_RW. This value is calculated when a file command is executed.


Modify the Arrays

The arrays that need modification are in the program scope:

- data_read
- data_to_write
- UDT CF_File_Structure

The third change to make is the definition of the UDT CF_File_Structure, shown in the following table.

Name	Data Type	Style	Description
File_Name	String140		This is the file name. Remember to add a file extension. For example .dat
String_Length	INT	Decimal	
File_Handle	DINT	Decimal	Do not manipulate this value
Bytes_to_RW	DINT	Decimal	Enter the number of bytes you want ot write to the file
Actual_Bytes_to_	DINT	Decimal	Do not manipulate this value
Chunks_to_RW	DINT	Decimal	Do not manipulate this value
Point_Method	DINT	Decimal	Do not manipulate this value
Pointer_Offset	DINT[2]	Decimal	Do not manipulate this value
Point_Move_Len	DINT	Decimal	Do not manipulate this value
Point_Last_Positi	DINT	Decimal	Do not manipulate this value
Commands	DINT	Decimal	Do not manipulate this value
Spare	DINT	Decimal	
Data	SINT[4950]	Decimal	This is the data that is written/read

 Data Element

Modify the Data Element

You must modify the Data element. This action modifies all usage of this data type throughout the application. The maximum size of the Data element is 65,000.

Notes:

Additional SD Card Resources

Appending Data

There is an additional method to access the CompactFlash/SD card on the Logix controller. The information can be obtained at the following sample code website:

[http://search.rockwellautomation.com/
search?site=sample_code&client=samplecode&output=xml_no_dtd&proxys
tylesheet=samplecode](http://search.rockwellautomation.com/search?site=sample_code&client=samplecode&output=xml_no_dtd&proxystylesheet=samplecode)

This method can be found by searching for the title 'CompactFlash and SecureDigital functions library'.

This method is similar to the program discussed in this manual with two significant exceptions. This access method can read/write to a specific portion of a file, and it can append data to the end of a specific file.

Notes:

A

access more data 55
appending data 59
applications of the memory card file system
12

B

build an example 33

C

checksum calculations 18
configure message instructions 24
controller scoped tags 51
controllers supported 14
Create a file 17
create a file 17
create a file on the memory card 25

D

data parsing 18
data types 21
delete a file on the memory card 32
delete command 20

F

format a CompactFlash card 41
format an SD card 45
format with a personal computer 39

L

limited life 14
Logix Designer application 45

M

maximum data per message 13
memory card enhancements 9
memory card features 10
memory card file system features 11
memory card reader 39
memory card types 9
message error codes 53
minimum file size/resolution 14
modify the arrays 57
modify the data element 57

O

overall status codes 52

P

performance data tables 50

R

read command 19
read data from the memory card 30
read to memory card 12
removal and insertion of memory card 13
restrictions of the file system 13
RIUP 13
RSLogix 5000 software 41

S

sample code 59
sample file compatibility 11
software version memory card features 10
status user-defined data type 51
storage capacity 9
Studio 5000 Logix Designer 7
supported software 9

T

third-party SD card 9

W

write command 18
write data to the memory card 28
write to memory card 12

Rockwell Automation Support

Use the following resources to access support information.

Technical Support Center	Knowledgebase Articles, How-to Videos, FAQs, Chat, User Forums, and Product Notification Updates.	https://rockwellautomation.custhelp.com/
Local Technical Support Phone Numbers	Locate the phone number for your country.	http://www.rockwellautomation.com/global/support/get-support-now.page
Direct Dial Codes	Find the Direct Dial Code for your product. Use the code to route your call directly to a technical support engineer.	http://www.rockwellautomation.com/global/support/direct-dial.page
Literature Library	Installation Instructions, Manuals, Brochures, and Technical Data.	http://www.rockwellautomation.com/global/literature-library/overview.page
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	http://www.rockwellautomation.com/global/support/pcdc.page

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete the How Are We Doing? form at http://literature.rockwellautomation.com/idc/groups/literature/documents/du/ra-du002_-en-e.pdf.

Rockwell Automation maintains current product environmental information on its website at <http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page>.

Allen-Bradley, Baseline, CompactLogix, ControlLogix, Logix5000, Rockwell Software, Rockwell Automation, and RSLogix 5000 are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication LOGIX-AP007C-EN-P - February 2016

Supersedes Publication LOGIX-AP007B-EN-P - March 2006

PN-328712

Copyright © 2016 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.