



...Display, Control, Communicate



Tutorial

microSD™ Card Functionality

Table of Contents

TABLE OF CONTENTS	2
INTRODUCTION.....	3
INSERTING THE MICRO SD CARD.....	3
SYSTEM MENU – REMOVABLE MEDIA	4
FLASH FILENAME FORMAT	5
SAVING AND LOADING PROGRAM FILES.....	7
CAPTURING SCREEN IMAGES.....	8
DATA LOGGING TO THE COMPACT FLASH	8
READ FUNCTION	9
WRITE FUNCTION	10
RENAME FUNCTION	11
DELETE FUNCTION	12
STATUS VALUES RETURNED BY THE FLASH MEMORY FUNCTIONS	12
<i>System Registers used with the flash memory card</i>	<i>13</i>
PROGRAMMING EXAMPLE – LOGGING A .CSV FILE	13
LADDER PROGRAMMING.....	14
SCREEN EDITOR PROGRAMMING.....	19

Introduction

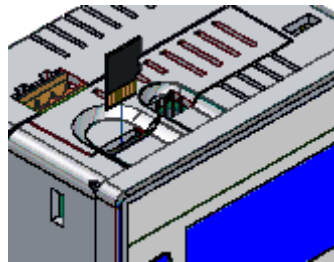
The i^3 controllers whose part numbers end with F are compact Flash card enabled. The slot marked memory will accept a microSD flash memory card. We would recommend that a SanDisk card be used. These cards come in various memory capacities ranging from 128MB to 2GB.

The μ SD card functionality allows the user to download their program from the i^3 to the card or upload a program from the card to the i^3 . This will enable customers to send out new programs on μ SD cards and have the end user up date the i^3 accordingly. Another feature is the data-logging function, which allows the user to log data registers from the i^3 into a .CSV format stored on the μ SD card. The μ SD card can then be inserted into a computers media slot and be opened directly using Microsoft Excel. Data can also be read back from a .CSV file into the i^3 registers. The μ SD card can also be used to capture screen shots, which can be helpful for diagnostics.

In this tutorial we will explore the functionality available with the μ SD card and program a small data-logging example.

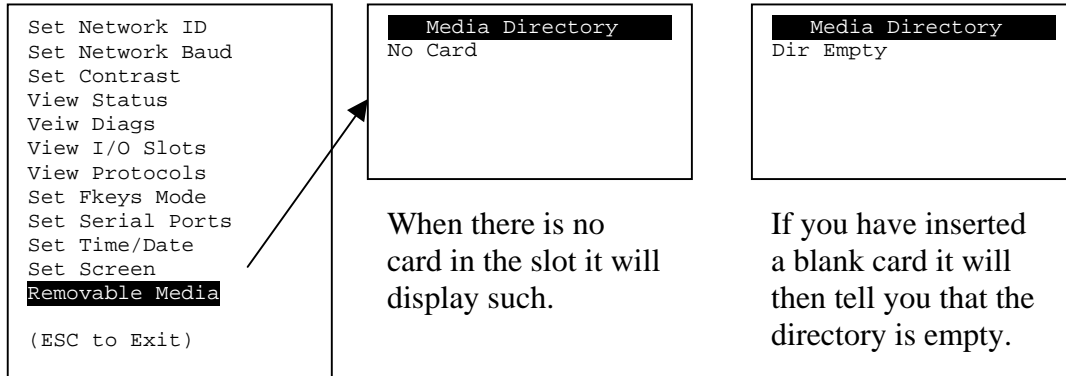
Inserting the micro SD card

The μ SD card can be inserted safely into the i^3 whether the unit is powered or not. The μ SD card slot on the i^3 controller is of the type push-in, push-out. To insert the card as shown below, point its 8-pin gold edge connector down, facing the front of the i^3 . Then gently push the card in at it will latch in. To remove, push the card and it will pop out.

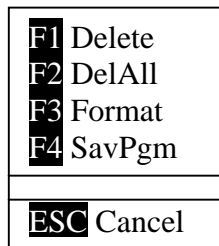


System Menu – Removable Media

To enter the *i*³ system menu press the up and down arrow together. Then use the arrow keys to move down to the Removable Media menu, press enter when it is selected

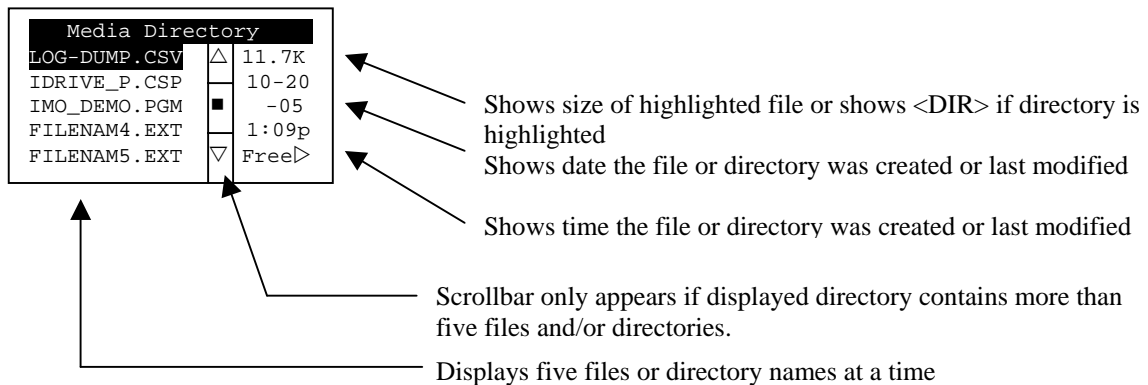


By pressing one of the soft-keys at the sides of the screen you will bring up another menu.



The numeric keys below the screen will carry out the function detailed. 1 will delete the selected file, 2 will delete all, 3 will format the card and 4 will save the program in the *i*³ to the μ SD card. The escape button will cancel and if pressed again exit the menu.

If there are files on the card then they will come up in a list on the left, with the selected file having its properties displayed on the right.



Flash Filename Format

The Flash functions support the flash with a Windows standard fat-16 file system. All names must be limited to the "eight dot three" format where the filename contains eight characters a period then a three-character extension. The entire filename including any path must be less than or equal to 147 characters.

When creating filenames and directories it is sometimes desirable to include parts of the current date or time. There are six special symbols that can be entered into a filename that are replaced by the format with current time and date information.

Symbol	Description	Example
\$Y	Substitutes the current 2-digit year	2004 = 04
\$M	Substitutes the current month with a 2-digit code	March = 03
\$D	Substitutes the current 2-digit day	22nd = 22
\$h	Substitutes the current 2-digit hour in 24-hour format	4pm = 16
\$m	Substitutes the current 2-digit minute	
\$s	Substitutes the current 2-digit second	
\$p	Substitutes the currently displayed 4-digit screen number (1-1023) (Intended mainly for screen capture)	Screen 53 = 0053

Note: All start with the (\$) dollar sign, and Date symbols are in upper case where as time symbols are in lower case.

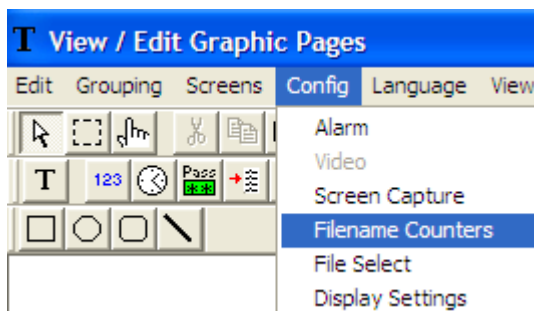
Examples of time/date stamped logged files (.csv)

Current date and time: April 24, 2007 3:45:34 PM

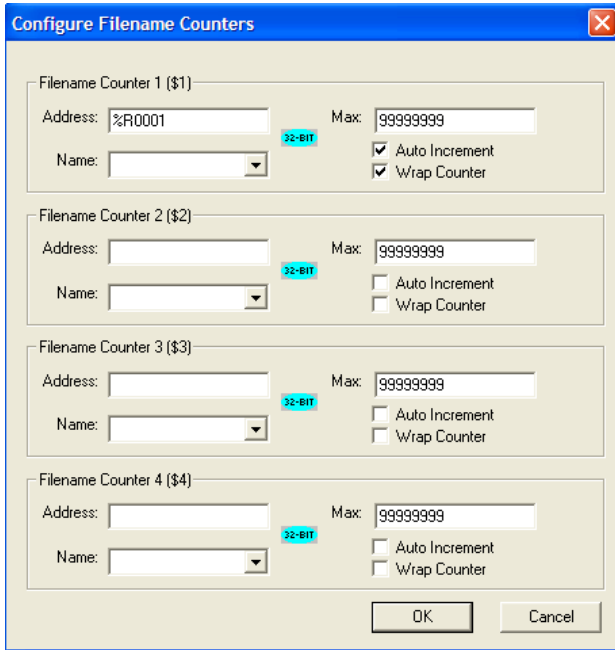
Filename in software	Filename as it appears
Data\$M\$D.csv	Data0424.csv
Year\$Y\Month\$M\aa\$D_\$h.csv	Year07\Month0\aa01_15.csv
Month_\$M\Day_\$D\\$h_\$m_\$s.csv	Month_04\Day_24\15_45_34.csv

Note: The file name extension **MUST** be included. It is not automatically added. ie. .csv for data logging files and .bmp for screen captures.

Another option to storing files in the flash is to use the Filename counters. This is an option in which a counter is incremented every time a file is stored to the flash memory card.



The filename counters can be configured in the screen editor Config menu.



Up to four filename counters can be configured. Two registers (in total 32bits) needs to be assigned for each register intended for use.

A maximum figure can be set and there are two options: Auto-increment, the counter is incremented every time and Wrap-counter, the filename count loops round.

The filename counters can be accessed wherever compact flash functions require a path name. Similar to the usage of the date and time symbols, filename counter usage is as follows:

$\$[\text{counter number}]u[\text{number of digits (1-8)}]$

For example, using counter 1 for a screen capture, if the counter has a Max value of 9999, the current value is 35 and the Auto Increment is checked:

$\$1u4 = 0035$

A combination of date/time and filename counters can be used in setting up the path and filename stored in the flash memory card.

Example the next time the screen is captured, the value will be 0036, then 0037, etc. and the current data and time is: Current date and time: April 24, 2007 3:45:34 PM,

Using counter 3 Auto Incrementing, Max of 59, currently at 58, Wrap turned ON

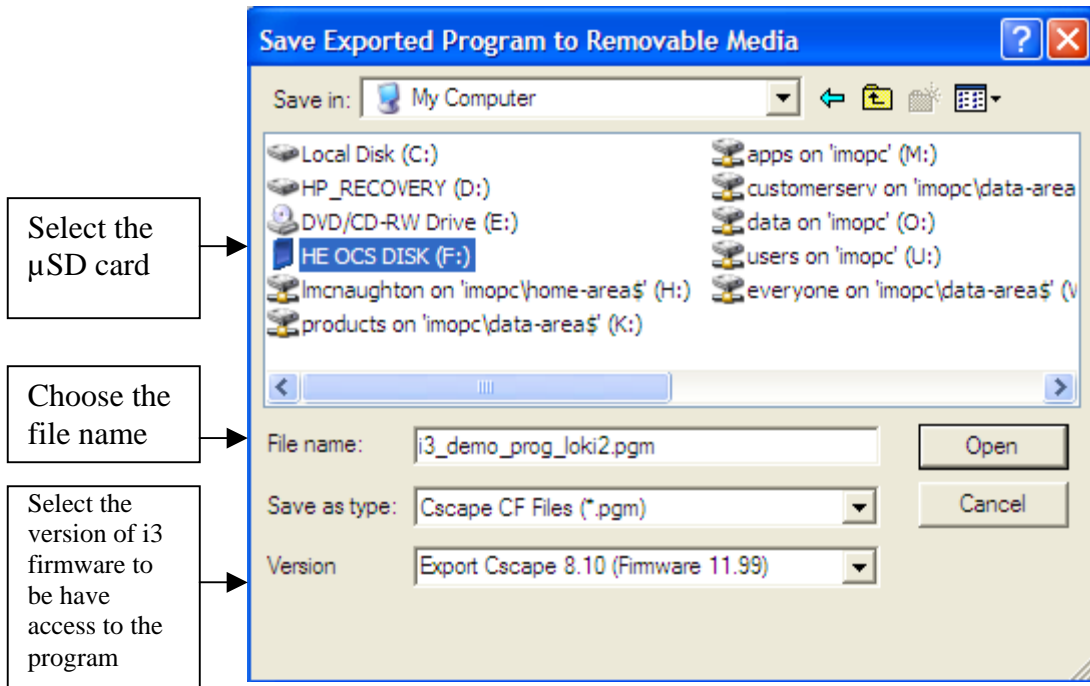
Filename in software	Filename as it appears
images\Chan3\SM-\$D-\$Y\\$h\$m-\$3u2.bmp	Images\Chan3\24-04-07\1545-58.bmp
Next screen snap shot (59)	Captures\Chan3\24-04-07\1545-59.bmp
Next screen snap shot (00, as wrap is enabled)	Captures\Chan3\24-04-07\1545-00.bmp

Note: The filename extension MUST be included as it is not automatically added.

Saving and Loading Program files

The μ SD card can be used to store program files on it, much as it could store music files, however if you wish to be able to upload those program files in the i^3 then you need to export them in the i^3 -configurator as a .pgm file instead of a .csp.

First you need to insert the μ SD card into your computer and then select the option from the File menu “Save Exported Program to Removable Media” in i^3 -configurator.



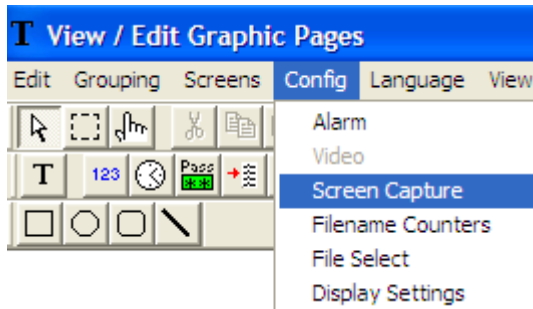
To load the application from Micro SD memory card to the i^3 , first you need to insert the card into the slot. Then use the Removable Media Manager to find and highlight the desired .PGM file, and then press ENTER.

To save a file to the card from the i^3 , insert the card and open the Removable Media Manager in the System Menu. Press the F4 function key (SavPgm), to save the application. The current program will be saved in a file called DEFAULT.PGM in the Micro SD root directory.

Note: Saving an application to the Micro SD card can only be done from the System Menu and is not available on a Removable Media Manager object that was placed on an application graphics screen by i^3 -configurator.

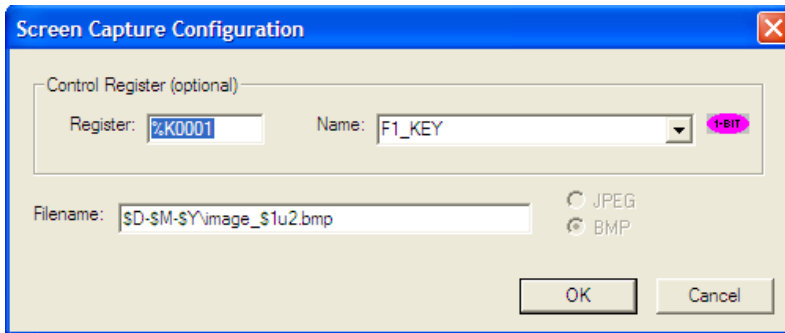
Capturing Screen Images

Within the *i*³ screen editor it is possible to select a control register in order to save bitmap images of the screen to the micro SD card. This is a very helpful tool for diagnostic reporting as an End user will be able to take a screen capture and email it to a support engineer. It can also be used to take captures of current trends for reporting.



In order to take screen captures we need to set up a controlling register to initiate the snap shot.

This is located in the Config menu of the screen editor.

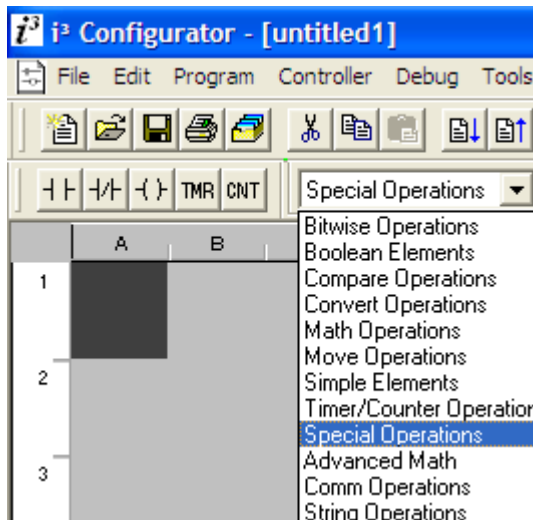


The controlling register needs to be a bit. It can be time based or event driven, i.e. User pressing a function key.

The file name follows the format of the compact flash filenames

Data Logging to the Compact Flash

The micro SD card can be used to store logged data in a comma-delimited file format, .csv. Using Read and Write Removable Media function blocks, an application ladder program can read and write *i*³ register data (%Rxxx) into this file format (.csv) that can be directly opened with standard database and spreadsheet PC programs (i.e. Microsoft Excel). Additional features include the functions: Rename and Delete.



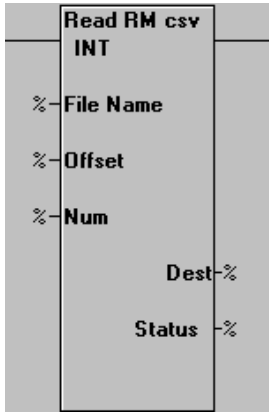
There are four compact flash ladder operations. These can be found in the special operations function menu.



- Read – Read CSV files into i3 registers
- Write – Write registers to flash memory
- Name – Rename files on the flash card
- Delete – Delete files on the flash card

Read Function

This function allows for data logged to the flash card in .csv format to be read back into the i^3 registers. This function requires to be powered until it is complete. The function is complete when the status returns a zero. If power is removed before the function is complete then it will error.



File Name: This can be a constant or a variable stored in a register. The name must follow the flash naming format. If stored in a register then the file the must be terminated will a NULL (byte containing zero). To indicate that you wish to indicate the name is in a register use %Rxxxx. Example: Data\dump_log.csv

Offset: Defines where to start reading the data. It can be a constant or a 32-bit register. Example: the data and offset in a csv file:

Offset	0	1	2
Data	123	456	789

Number of Elements:

Determines the number of elements to be read. It can be a constant or 16-bit register. For ASCII data this becomes "Max Number of Characters" and sets the maximum number of characters that can be read from the file and stored in controller registers. Should a comma (not in quotation marks) be encountered before the Maximum Number of Characters has been read, then the function will stop reading characters instead of continuing to the Maximum Number of Characters specified.

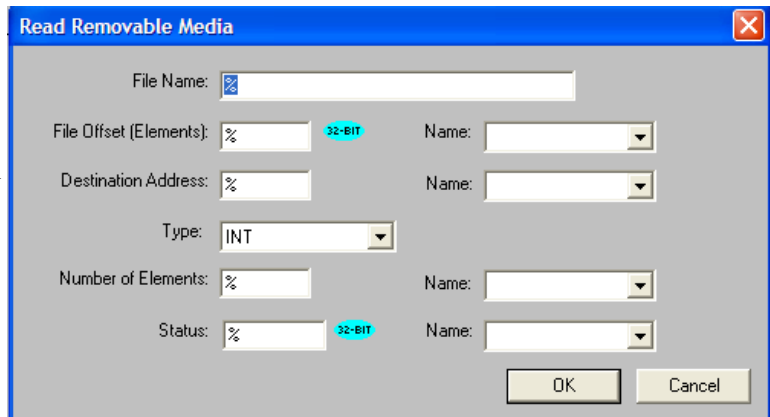
For Boolean data it is always set to one.

Destination Address:

Controls where the read data is stored. This can require large amounts storage (%Rxxx) with each element possibly requiring more than one register (DINT, REAL, UDINT, ASCII types)

Type:

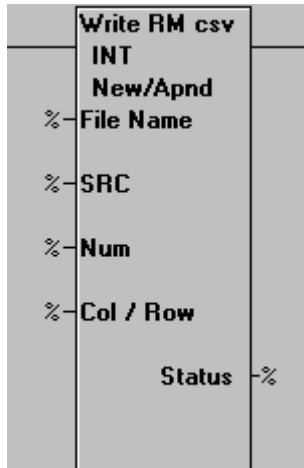
This is the type of data that is to be read. The .csv file doesn't have this information encoded in it and therefore is the programmer responsibility.



Status: Displays the status codes for the process

Write Function

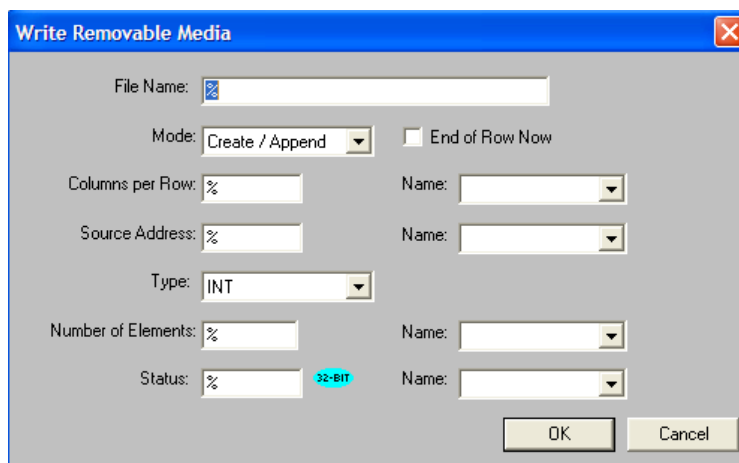
This function allows for the creation and appending of .csv files to the flash card. These files can be directly opened in Spreadsheet packages (Microsoft Excel) and is therefore very useful for report creation. This function requires to be powered until it is complete. The function is complete when the status returns a zero. If power is removed before the function is complete unlike the read function it will complete its operation.



Filename: This follows the same format as the read function.

Mode: 4 of
Create – creates a new file. If the file exists, then an error will occur.
Append – appends an existing file, if there is not one then it will error.
Create/Append – Hybrid of the two above. (Best to use this mode)
Overwrite – if the file exists overwrite with a new file

Source address: Starting register in which to log from.
Type: Type of data to be logged.
Number of Elements: This is just the same as the read functions parameter. It determines how many elements are to be written.
Status: This displays the status codes of the function.



End of Row Now: Setting this option will cause the row to end, at the end of the write function. This operates in conjunction with Columns per Row.

Columns per Row: Defines the format for writing data into the .csv file. This can be a constant or 16-bit register.

When a .csv file is written to in a table format it can be viewed in a column / row format like a spreadsheet. Setting this parameter determines how many elements to write in a single row before a new row is started.

Setting this value to zero will disable the generation of new rows and will generate all data as a single row.

Examples:

3 columns per row

1	2	3
4	5	6
7	8	9

5 columns per row

1	2	3	4	5
6	7	8	9	10

Rename Function

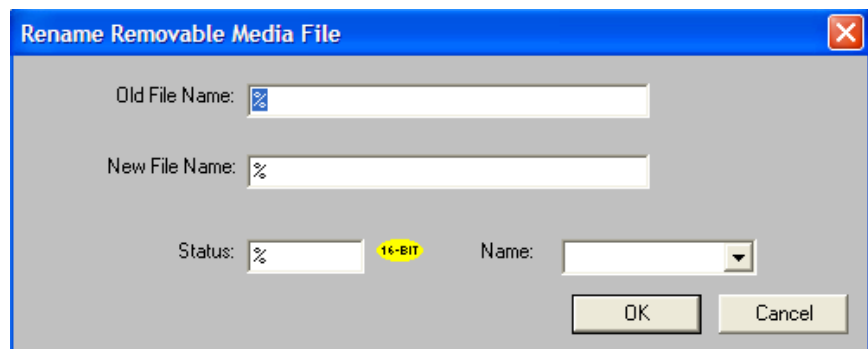
This function allows for the renaming of existing files. On enabling power to this function it will start, if power is removed before it has finished, it will still complete its operation.

Rename RM	
%	Old Name
	New Name %
	Status %

Old Name: The name of the existing file follows the same naming format; the same as the previous functions.

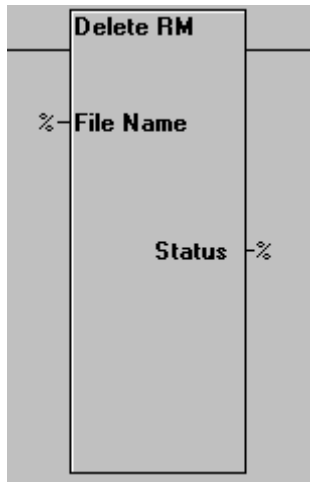
New Name: This is the new name for the file, and has the same parameters as above.

Status: Displays the status of function.



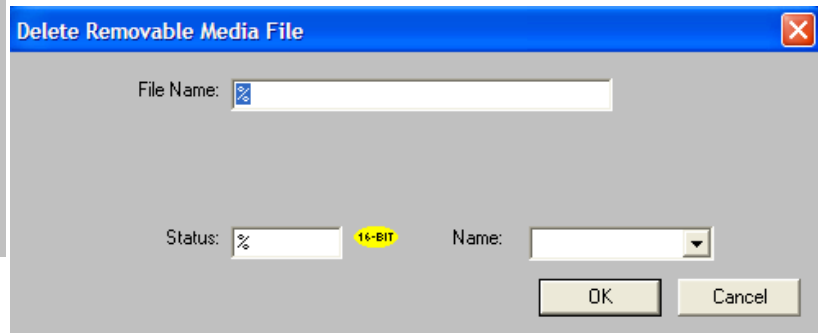
Delete Function

This function deletes a file from the flash memory card. On enabling power to this function it will start, if power is removed before it has finished, it will still complete its operation.



File Name: The name of the file to delete follows the naming format as the previous functions.

Status: Displays the status of function.



Status Values Returned by the flash memory functions

Value	Description
0	Operation completed successfully
-1	End of file was reached before completing
-2	Function is active, waiting for operation to complete
-3	Function is waiting on another CF function to complete
-4	Function block is inactive (usually no power flow)
1	Card present but unknown format
2	No card in slot
3	Card present, but not supported
4	Card swapped before operation was complete
5	Unknown error
66	File / Path specified does not exist
73	Bad file descriptor (corrupt file)
77	Attempt to open / rename file that is open
81	Specified file already exist
86	Function block contains illegal parameter
88	Too many open files*
92	Attempt to write failed

* - These Status Codes *should* occur, being duplicates of higher priority status codes. However, they are included in the code because some people can break anything.

94	Sharing violation*
95	No disk present*
96	Directory structure corrupt
98	Incorrect data format

System Registers used with the flash memory card

%SR175 Status: This shows the current status of the Flash interface.

Possible status values:

0	CompactFlash Interface OK
1	Card Present but unknown format
2	No card in slot
3	Card present, but not supported
4	Card swapped before operation was complete
5	Unknown error

%SR176 Free Space: This 32-bit register shows the free space on the CompactFlash card in bytes.

%SR178 Card Capacity: This 32-bit register shows the total card capacity in bytes.

Programming Example – Logging a .CSV file

In this example we are going to utilise the two temperature inputs, as well as flash card capabilities in the model i3A12X/13C14-SOHF.

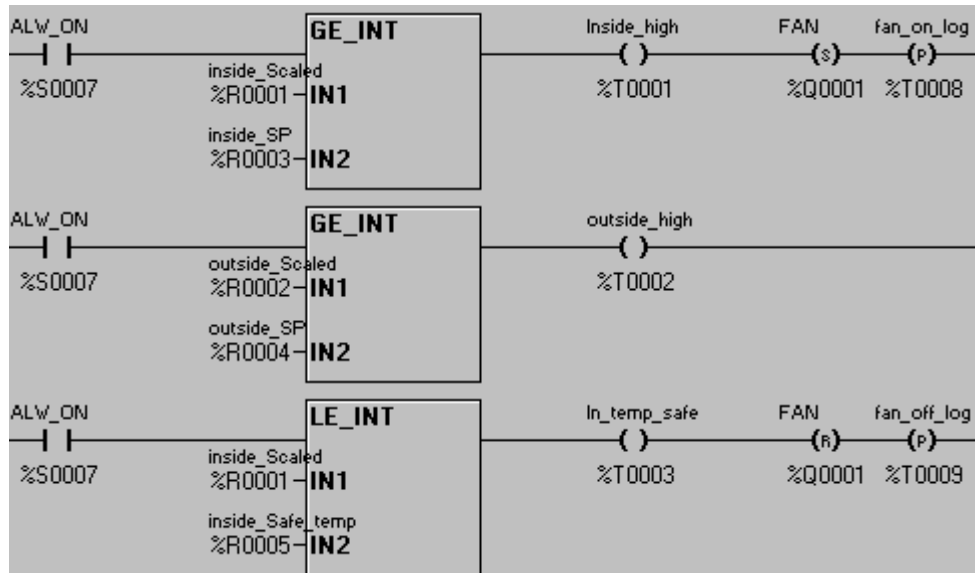
We are going to measure a temperature inside and outside of a controlled environment. If the temperature on the inside exceeds a set limit then we will operate a fan via a digital output to cool the inside back down to a safe level.

There will be a door on the controlled environment allowing access, with a sensor to monitor when the door is opened. This digital input sensor that requires calibrating, of which the process requires the following procedure: Hold the sensor on for 4s, after 4 seconds release, then hold the sensor on the 1s, before releasing. If the sensor output is maintained on then the sensor has been calibrated properly, if not then re-calibrate.

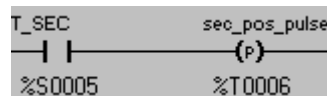
We want to log the temperature every hour, log when the digital input has been trigger, log when the fan has had to be operated and for how long. A new log must be created every day, with the date in the title of the filename.

Now that we have scaled the values we can program the compare logic required to operate the fan and operate the compact flash write functions for logging.

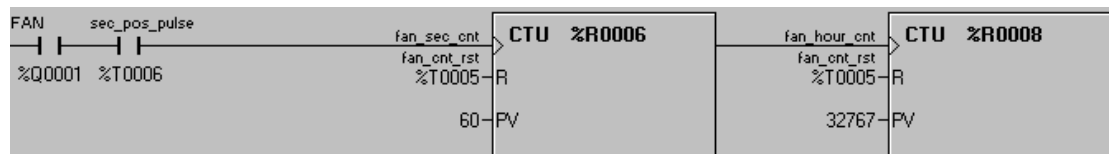
If the inside value is greater or equal to (GE) a user specified set point then set the fan on and operate a bit. The outside temperature also requires this logic however we do not need to switch on the fan. To switch off the fan, the inside temperature must reduce to a user specified safe temperature.



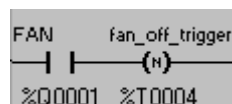
In order to ensure that we only count once per second pulse we use a positive edge transition operated by the *i*³'s internal RTC second pulse.



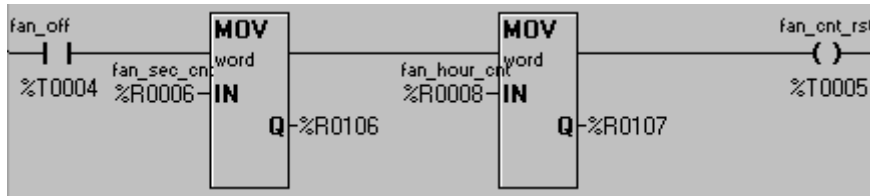
For the fan on count we use the second positive edge pulse anded with the fan output itself. This then operates a counter function counting seconds which outputs to a counter that counts the hours.



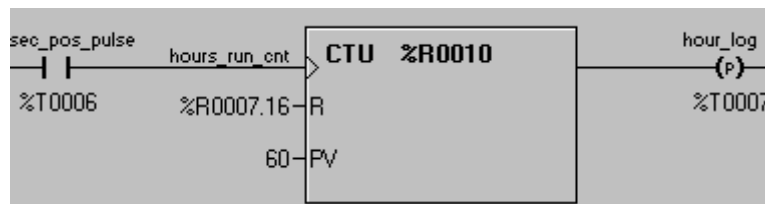
When the fan switches off it triggers an output bit (negative edge) that moves the time elapsed whilst the fan was on, into registers to be logged.



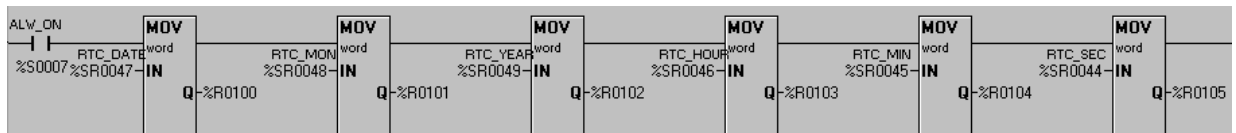
The registers are consecutive with the rest of the data we want to log to the microSD card.



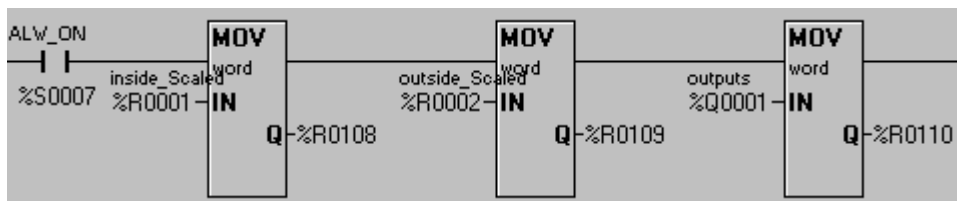
We also want to log every hour, so by reusing the positive edge second pulse we count up the minutes and trigger an output bit when the hour has been made. This output bit triggers a compact flash write function.



We need to move the *i*³'s RTC data from system registers to consecutive register locations next to our data we wish to log. To do this we will use move functions that are always active.



We also need to move the registers we set up to store the scaled analogue input values and the state of the outputs to the consecutive register area that we intend to log.



Now that we have all the data we wish to log in a consecutive register area (%R100 - %R110) we can enter the logic to operate the Compact Flash write function.

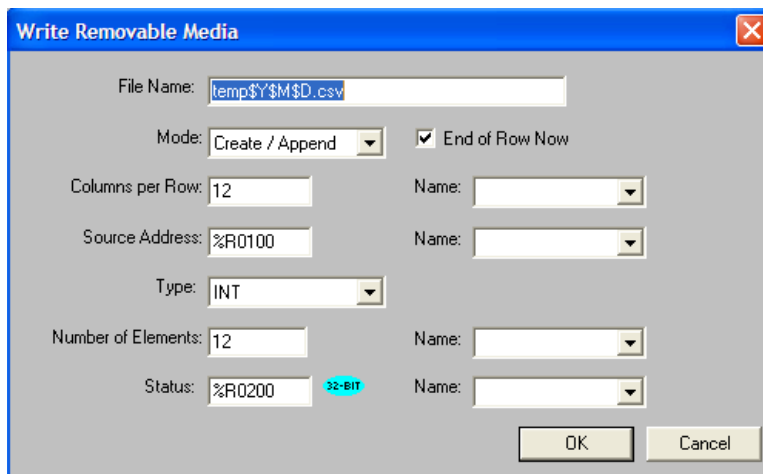
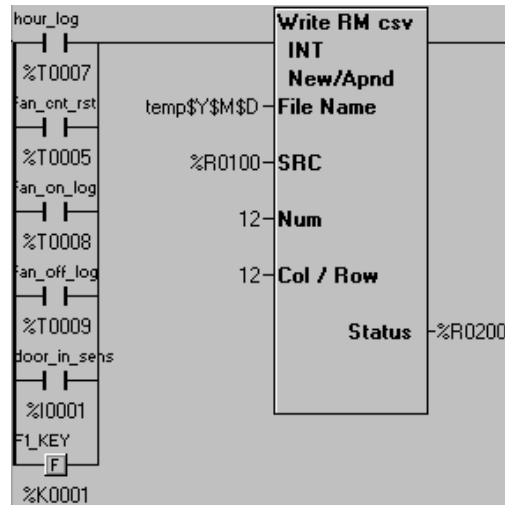
The trigger for the write function will be:

- Ever hour action
- Fan operation
 - ON
 - OFF
- Digital input triggered
- User key for debugging

Enter the filename as

temp\$Y\$M\$D.csv

This follows the filename format and the file will appear in the microSD card as temp070327 (temp then followed directly by the current date).

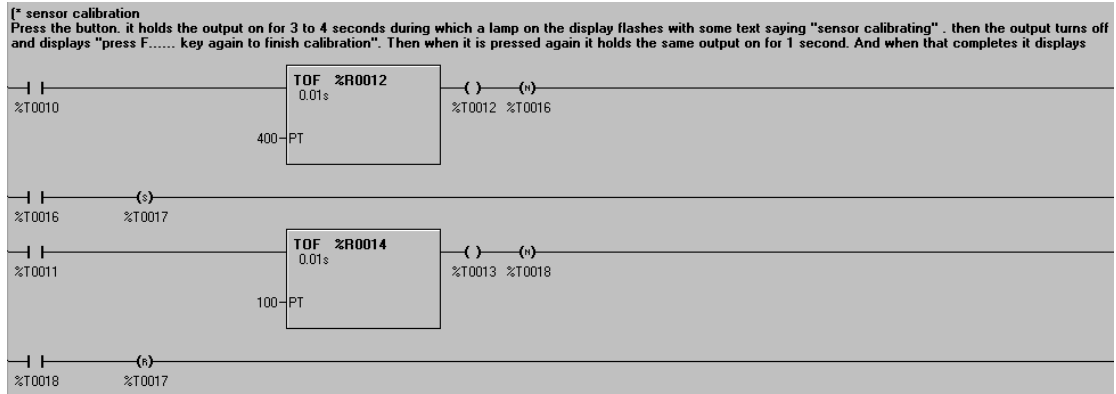


Ensure that the mode is Create/Append, with the end of row now box ticked.

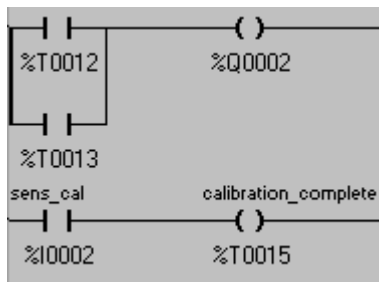
Set the rest of the parameters as shown.

As we are using a digital input sensor that requires some tuning we can automate this procedure using some code. The sensor has a second output to indicate when it is calibrated. Following the specification we need to write the following ladder logic code.

MicroSD Card

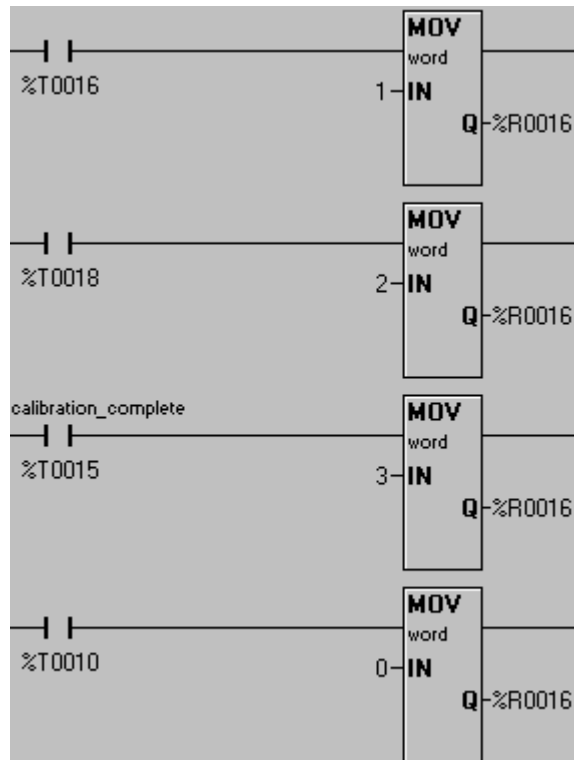


The user need to operate a bit and have the output on for 4 seconds then release and prompt the user to operate another bit that will switch the output on for 1 second.



The calibration will be complete after these two steps have been completed. If the calibration has been successful then the input will be on, if not then it will be off.

We will use a text table function in the screen editor to display the status of the calibration process. Hence we need to use move functions to move corresponding values into a register to be used.

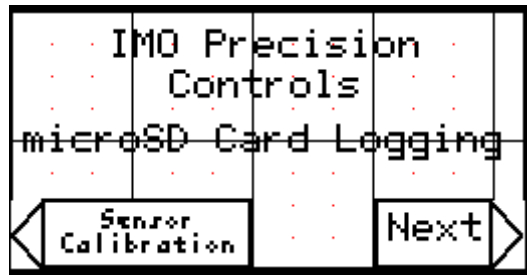


Screen Editor Programming

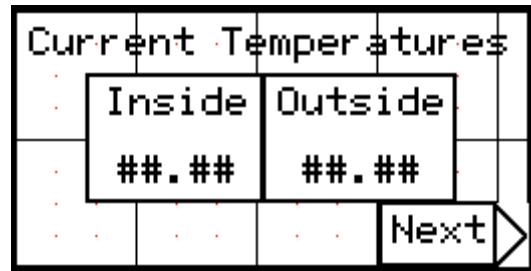
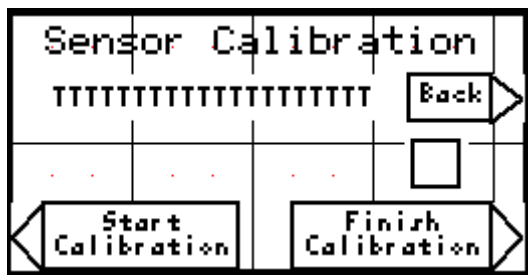
Now that the ladder logic has been program we can program some HMI screens for user interaction.

From the main screen the user will have two options, to screen jump to the sensor calibration screen or continue to the next screen.

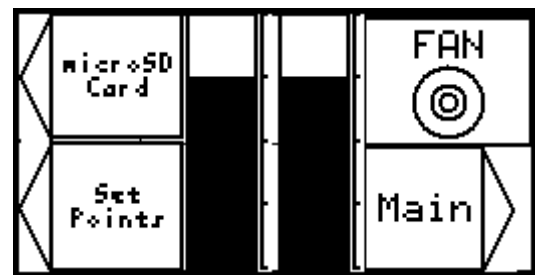
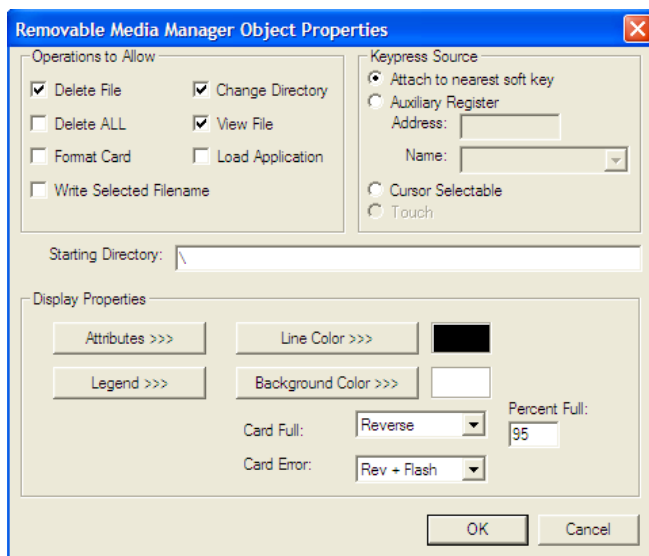
In the sensor calibration menu the user will have a text table displaying the stage of calibration. A lamp to indicate when it is calibrating and two switches to initiate the calibration stages



Pressing next takes the user to an intermediate screen that displays the two temperatures and a screen jump to the system menu.



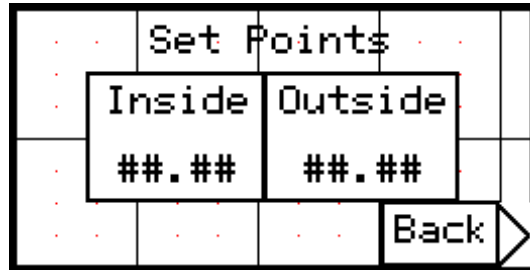
In the main menu there are two bar graphs for displaying the inside and outside temperatures and a lamp to indicate when the fan is on.



There are two screen jump buttons, one to take you back to the main entry screen and the other to take the user to the set point entry screen.

Also on the screen will be a Media Manager function . This function allows the user to view the media card and provides them with an easy access menu to change files on the card without going through the *i*³ menu system menus.

The last screen is the Set Points screen which will have two numeric data functions that will allow the user to enter the Inside and Outside set points. There is also a screen jump function to allow the user to go back to the last screen.



Please refer to the *i*³ Configurator program: i3_tut_microSD_log.csp





IMO Precision Controls Limited
 1000 North Circular Road
 Staples Corner, London
 NW2 7JP United Kingdom
 Tel: +44 (0)20 8452 6444
 Fax: +44 (0)20 8450 2274
 Email: imo@imopc.com
 Web: www.imopc.com



IMO Jeambrun Automation SAS
 165 Rue Jean Jaures,
 94700 Maisons Alfort
 Paris, France
 Tel: +33 (0)1 45 13 47 05
 Fax: +33 (0)1 45 13 47 37
 Email: info@imopc.fr
 Web: www.imopc.fr



IMO Deutschland
 Für weitere Einzelheiten
 zu IMO Agenten und
 Distributoren in Ihrer Nähe
 schreiben. Sie bitte ein E-mail
 an folgende Adresse:
 imo@imopc.com



IMO Italia
 Viale A. Volta 127/a
 50131 Firenze, Italia
 Tel: +39 800 783281
 Fax: +39 800 783282
 Email: info@imopc.it
 Web: www.imopc.it



IMO Canada
 Unit 10, Whitmore Road
 Woodbridge, Ontario.
 L4L 8G4 Canada
 Tel: +1 905 265 9844
 Fax: +1 905 265 1749
 Email: imocanada@imopc.com



Cam Switches
 Din Terminals
 Drives
 Enclosures
 Fieldbus remote I/O
 Isolators & Switch Fuses
 MCB & RCD
 Motor Circuit Breakers
 Motor Control Gear
 Panel Meters
 Relays
 Signal Conditioning
 Sockets
 Timers
 Transformers &
 Power Supplies



Drives
 Intelligent Terminals/HMI
 Limit Switches
 Photoelectric Switches
 PLCs
 Proximity Switches
 Temperature Controls



Data Acquisition & Control
 Drives
 Intelligent Terminals/HMI
 Limit Switches
 Photoelectric Switches
 Proximity Switches
 PLCs
 Signal Conditioning
 Temperature Controls



Lightguards
 Safety Limit Switches
 Safety Relays

All IMO products are tried, tested and approved
 to relevant international quality standards



Jaguar VXM 0.37-500KW
 Jaguar VXSM 0.37-7.5KW
 Jaguar CUB 0.37-2.2KW



Audible devices
 Chip-on-Board
 Device programmers
 LEDs & 7 seg. displays
 PCB Terminal blocks
 Relays - automotive
 Relays - power
 Relays - signal
 Switches

