Sommario

DEVICE DATACELL

1. Introduction

The DATACELL device, is used to store data in a non-volatile device and is inserted in the family Micro-Qmove when there is no HMI device;
DATACELL has:

- flexible memory management programs;
- a programmable memory access from QCL in writing and reading;
- an insertion of an end-of-program for each program.

1.1 Installation

1.1.1 DEVICE DECLARATION IN THE CONFIGURATION FILE (.CNF)

In the configuration unit (.CNF), the BUS section must be declared so that you have the hardware resources required for the implementation of the DATACELL device.

```
;---------------------------------
; Internal device declaration
;---------------------------------
INTDEVICE
<device name>       DATACELL         TCamp
```

where:

<device name>	The name assigned to the device.
DATACELL	Keyword that identifies the device of recipe management.
Tcamp	Time sampling device (1÷250 ms).

1.1.1.1 Example

```
;--------------------------------
; Internal device declaration
;--------------------------------
INTDEVICE
dcData      DATACELL      0004
```

1.2 Operation

The DATACELL device used to access a memory area equal to 8192 long (32768 bytes) available in the non-volatile memory of the instrument. This memory area is designed to be used as a data container for work programmes or recipes.
The device allows you to arrange access to these data through a table with rows (in numbers equal to a “numstep” parameter) and columns (in numbers equal to a “numprog”). In each box of this table you can set up a number of variables of type long from 1 to 6 (in the “numelem” parameter). The user of the device must set the value of the “numelem” and “numstep” parameters and as a result the device counts the number of columns that make up the table and writes in the “numprog” parameter.

The formula that the device uses is:
numprog = 8192 / (numstep * numelem + [1])

The presence of the number [1] depends on the setting of the bit 0 of the “prgsetting” parameter. The 0 bit of the “prgsetting” parameter allows you to assign each program a variable to indicate which is the step to end program.
If for example, we want to create a table to hold the work programmes with 10 steps each and every step with 4 variables, we can have a total number of programs of 204. If you want to assign to each program a variable to indicate which is the step to end program, the number of programs will be 199.
To write the data into the table that is created you have to use the WRITESTEP command. Before using this command, you must set the “progin” and “stepin” parameters the coordinate of the box that you want to write (in progin is write the column and in stepin the row). Also you have to write the values to be transferred into the box in the “elema” … “elemf” parameters. At this point you can store these data in memory by sending WRITESTEP command. The same system is used to write the variable to indicate the step of fineprogramma using the “elemend” parameter.

1.2.1 Example of writing:

```
[...]
dcData:progin = 2                       ;work program n°2
dcData:stepin = 7                       ;step 7 of the program
dcData:elema = 10                       ;first variable
dcData:elemb = 45678                    ;second variable
dcData:elemc = 12                       ;third variable
dcData:elemd = 345678768                ;fourth variable
dcData:stepout = 0
```

```
WRITESTEP dcData                        ;send write command
WAIT dcData:stepout EQ dcData:stepin    ;wait command executed
[...]
```

To read data from memory, use the READSTEP command. Before using the command you have to set the “progin” and “stepin” parameters the coordinate of the box that you want to read (in progin writing the column and in stepin the row). You can now read the data into memory by sending the READSTEP command. Data read will appear in the “elema” ... “elemf” and “elemend” parameters.

1.2.2 Example of reading:

```
[...]
dcData:progin = 2                       ;work program n°2
dcData:stepin = 7                       ;step 7 of the program
dcData:stepout = 0
READSTEP dcData                         ;send read command
WAIT dcData:stepout EQ dcData:stepin    ;wait command executed
glVar01 = dcData:elema                  ;first variable accessed
glVar02 = dcData:elemb                  ;second variable accessed
glVar03 = dcData:elemc                  ;third variable accessed
glVar04 = dcData:elemd                  ;fourth variable accessed
[...]
```

1.3 Commands and parameters table

1.3.1 SYMBOLS USED

The **name** of the parameter, state or command are shows on the left of the table.

R
Indicates if the parameter or state is retentive (upon initialization of the device maintains the previously defined), or the state assumes upon initialization of the device.
If the device does not need to initialize the field “R” indicates the value that the parameter or state take at the power up of the card.
R = Retentive
0 = Upon initialization of the device the value is forced to zero.
1 = Upon initialization of the device the value is forced to one.
- = Upon initialization of the device is presented significant value.

D
Indicates the **dimensions of the parameter**.
F = Flag
B = Byte
W = Word
L = Long
S = Single Float

1.3.1.1 Conditions

Are desscribe all the **conditions that must exist is considered correct or because the command is accepted**.
In some cases, limit values are specified for the acceptance of the parameter: If there are any values outside the limits set, the data is in any case accepted; therefore appropriate controls of the application must be provided to ensure the proper functioning.
To run a command, all the conditions required to be met; otherwise the command is not sent.

A
Indicates the access mode.
R = Read.
W = Write.
RW = Read / Write.

1.3.2 Parameters

Name	D	R	A	Conditions	Description
numelem	B	R	RW	-	**Elements number** Indicates the number of elements within a step Valid range: 1 ÷ 6
numstep	W	R	RW	-	**Step number** Indicates the number of steps in each program Valid range: 1 ÷ 8192

Name	D	R	A	Conditions	Description
numprog	W	-	R	-	**Program number** Indicates the number of the available programs. The value depends on 1) the total number of long available in program memory, 2) from the value of numelem parameter, 3) from the value of numstep parameter 4) by enabling the end of program. The formula is as follows: numprog = 8192 / [1].
progin	W	0	RW	-	**Program input** Indicates the program number to be stored with the WRITESTEP command or read with the READSTEP command.
stepin	W	0	RW	-	**Step input** Indicates the number of the step to be stored with the WRITESTEP command or read with the READSTEP command.
stepout	W	0	RW	-	**Step output** Indicates that the written step has been stored or the step into reading is available. To verify that the command sent (WRITESTEP or READSTEP) was run you should check that stepin = stepout.
elema..f	L	0	RW	-	**Elements A..F** Are the values of the step used with READSTEP and WRITESTEP commands
elemend	W	0	RW	-	**Elements for end program** If non-zero indicates that the step has end program.
readstep	-	-	R	-	**ReadStep** Reads the selected step in stepin
writestep	-	-	R	-	**WriteStep** Writes the selected step in stepin
prgsetting	B	R	RW	-	**Setting program data-entry** The ZERO bit enables the introduction of program end. When this bit is 1 the number of programs “numprog” becomes: nuprog = 8192 / (numstep * numelem + 1). If the bit “0” is a “0” the number of programs becomes: numprog = 8192 / (numstep * numelem)

1.3.3 Commands

Name	D	R	A	Conditions	Description
readstep	-	-	R	-	**ReadStep** Reads the selected step in stepin
writestep	-	-	R	-	**WriteStep** Writes the selected step in stepin

1.4 Limitations

The write operation via the WRITESTEP command should be done keeping in mind that because of the memory component used (serial Flash Eprom) this operation is costly in terms of time spent. In fact the time used is variable from 512 to 1024 times the sampling time associated with DATACELL device. So this type of memory can be used to contain data that can be changed by the operator with relatively slow times. Definitely not a usable memory to contain data that must be written with a high frequency. In any case the write operation is performed with a background mode and does not affect the performance of the CPU to handle the rest of the device and the application.
For example, if the sampling time associated with the device is 6 ms, the time to perform a write to device can range from approximately from 3 to 6 seconds. Stepout parameter becomes equal to stepin after this time.
Also the type of memory used guarantees a number of Scriptures equal to 100000. Even so you should avoid writing programs which they write continuously on memory using the WRITESTEP command.

[1] numstep * numelem) + 1 (when enabled the program end