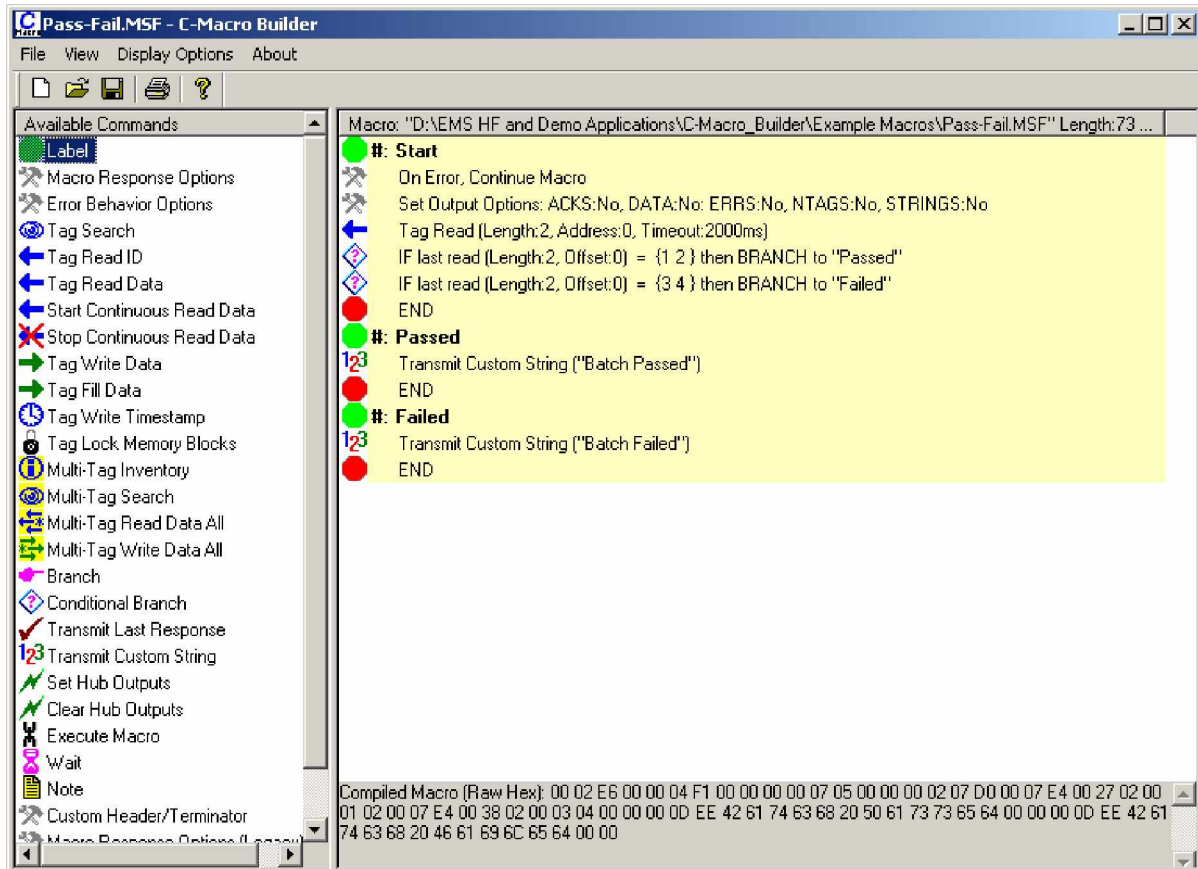


## ESCORT MEMORY SYSTEMS

# C-MACRO BUILDER

*Software Utility for Cobalt HF RFID Devices*



## USER'S GUIDE

*How to Use Escort Memory Systems  
C-Macro Builder Software Utility*



Escort Memory Systems reserves the right to make modifications and improvements to its products and/or documentation without prior notification. Escort Memory Systems shall not be liable for technical or editorial errors or omissions contained herein, nor for incidental or consequential damages resulting from the use of this material.

The text and graphic content of this publication may be used, printed and distributed only when all of the following conditions are met:

- Permission is first obtained from Escort Memory Systems.
- The content is used for non-commercial purposes only.
- Copyright information is clearly displayed (*Copyright © 2007, Escort Memory Systems, All Rights Reserved*).
- The content is not modified.

The following are trademarks and/or registered trademarks of Escort Memory Systems, a Datalogic Group Company: Escort Memory Systems®, and the Escort Memory Systems logo, Subnet16™, Cobalt HF™, RFID AT WORK™, C-Macro™, C-C-Macro Builder™, CBx™ and RFID Dashboard™.

Third party product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.



## C-Macro Builder ▪ User's Guide

P/N: 17-1325 REV 02 (05/07)

C-Macro Builder software is provided by Escort Memory Systems (EMS) "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed.

In no event shall EMS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages however caused and on any theory of liability, arising in any way out of the use of this software.



**DATALOGIC**  
Your Life. Our Enthusiasm™



**DeviceNet**  
Modbus-IDA  
the backbone for distributed automation

# CONTENTS

CONTENTS .....	3
CHAPTER 1: C-MACRO BUILDER OVERVIEW .....	5
<b>1.1 Using C-Macro Builder .....</b>	<b>6</b>
1.1.1 Building Macros .....	7
1.1.2 Saving Macros .....	8
<b>1.2 Loading Macros into an RFID controller .....</b>	<b>10</b>
CHAPTER 2: C-MACRO BUILDER COMMANDS.....	11
<b>2.1 Command List Icons .....</b>	<b>11</b>
2.1.1 Label .....	12
2.1.2 Macro Response Options.....	13
2.1.3 Error Behavior Options .....	16
2.1.4 Tag Search.....	17
2.1.5 Tag Read ID.....	18
2.1.6 Tag Read Data .....	19
2.1.7 Start Continuous Read Data .....	20
2.1.8 Stop Continuous Read Data .....	21
2.1.9 Tag Write Data .....	22
2.1.10 Tag Fill Data.....	24
2.1.11 Tag Write Timestamp .....	25
2.1.12 Tag Lock Memory Blocks.....	26
2.1.13 Multi-Tag Inventory.....	27
2.1.13.1 Anti-Collision Mode.....	27
2.1.13.2 Family Code (AFI).....	27
2.1.13.3 Tag Limit .....	27
2.1.13.4 Timeout (ms) .....	28
2.1.14 Multi-Tag Search .....	29
2.1.14.1 Anti-Collision Mode .....	29
2.1.14.2 Family Code (AFI).....	29
2.1.14.3 Tag Limit .....	29
2.1.14.4 Timeout (ms) .....	30
2.1.15 Multi-Tag Read Data All.....	31
2.1.15.1 Anti-Collision Mode .....	31
2.1.15.2 Start Address.....	31
2.1.15.3 Read Length (Bytes).....	31
2.1.15.4 Family Code (AFI).....	31
2.1.15.5 Tag Limit .....	32
2.1.15.6 Timeout (ms) .....	32
2.1.15.7 Include Tag ID with Data (Checkbox).....	32
2.1.16 Multi-Tag Write Data All.....	33
2.1.16.1 Write Data .....	33
2.1.16.2 Anti-Collision Mode .....	33
2.1.16.3 Start Address.....	33
2.1.16.4 Write Length (Bytes): .....	33
2.1.16.5 Timeout (ms) .....	33
2.1.16.6 Family Code (AFI).....	34
2.1.16.7 Tag Limit .....	34

2.1.17	Branch.....	35
2.1.18	Conditional Branch .....	36
2.1.18.1	Branch Conditions.....	36
2.1.18.2	Compare Data .....	37
2.1.18.3	Branch to Label.....	37
2.1.19	Transmit Last Response .....	40
2.1.20	Transmit Custom String .....	41
2.1.21	Set Hub Outputs.....	42
2.1.22	Clear Hub Outputs.....	43
2.1.23	Execute Macro .....	44
2.1.24	Wait.....	45
2.1.25	Note .....	46
2.1.26	Custom Header/Terminator .....	47
2.1.27	Macro Response Options (Legacy).....	48
2.1.28	End .....	49

## CHAPTER 1: C-MACRO BUILDER OVERVIEW

“**C-Macro Builder**” is a software tool designed by EMS that allows the user to quickly and easily create “*RFID Command Macros*”, which are simple, yet powerful, sets of instructions that can be loaded into EMS’ Cobalt and HF-0405 Series RFID controllers.

When executed, a macro can instruct an RFID controller to perform various operations. Macros can instruct the controller to perform any of the following:

- Write data
- Read data
- Fill or clear a tag
- Compare the results of read data
- Transmit custom string information based on the results of logical comparisons of the data
- Instruct a Subnet16 Hub to set or clear “Outputs”

The C-Macro Builder utility contains an intuitive drag-and-drop interface that displays the macro as a set of single comprehensive instructions, which allows the user to create their own custom macros using a very simple programming language.

Once a macro is created, C-Macro Builder saves the “compiled” string of byte values, which can then be downloaded to a Cobalt controller via the *Cobalt HF Dashboard* program (available online at [www.ems-rfid.com](http://www.ems-rfid.com)).

Cobalt controllers can store up to eight macros (numbered 1 through 8). Macros can be activated by a software trigger, a tag presence trigger, a “*continuous*” trigger or from a Subnet16 Hub Input trigger. Each controller can store up to eight triggers, which can be used to activate any of the eight stored macros.

Because macros also have the ability to execute other macros, they may be chained together to create even larger programs.

**NOTE:** For more information on configuring macro triggers, refer to the *Cobalt HF Serial Dashboard – User’s Manual*, publication P/N: **17-1335** (for serial connections) or the *Cobalt HF TCP/IP Dashboard – User’s Manual*, publication P/N: **17-1336** (for TCP/IP and Ethernet-based connections)

## 1.1 USING C-MACRO BUILDER

The C-Macro Builder user interface window is divided into three panes, the “*Command List*,” the “*Macro Workspace*,” and the “*Macro Data*” pane.

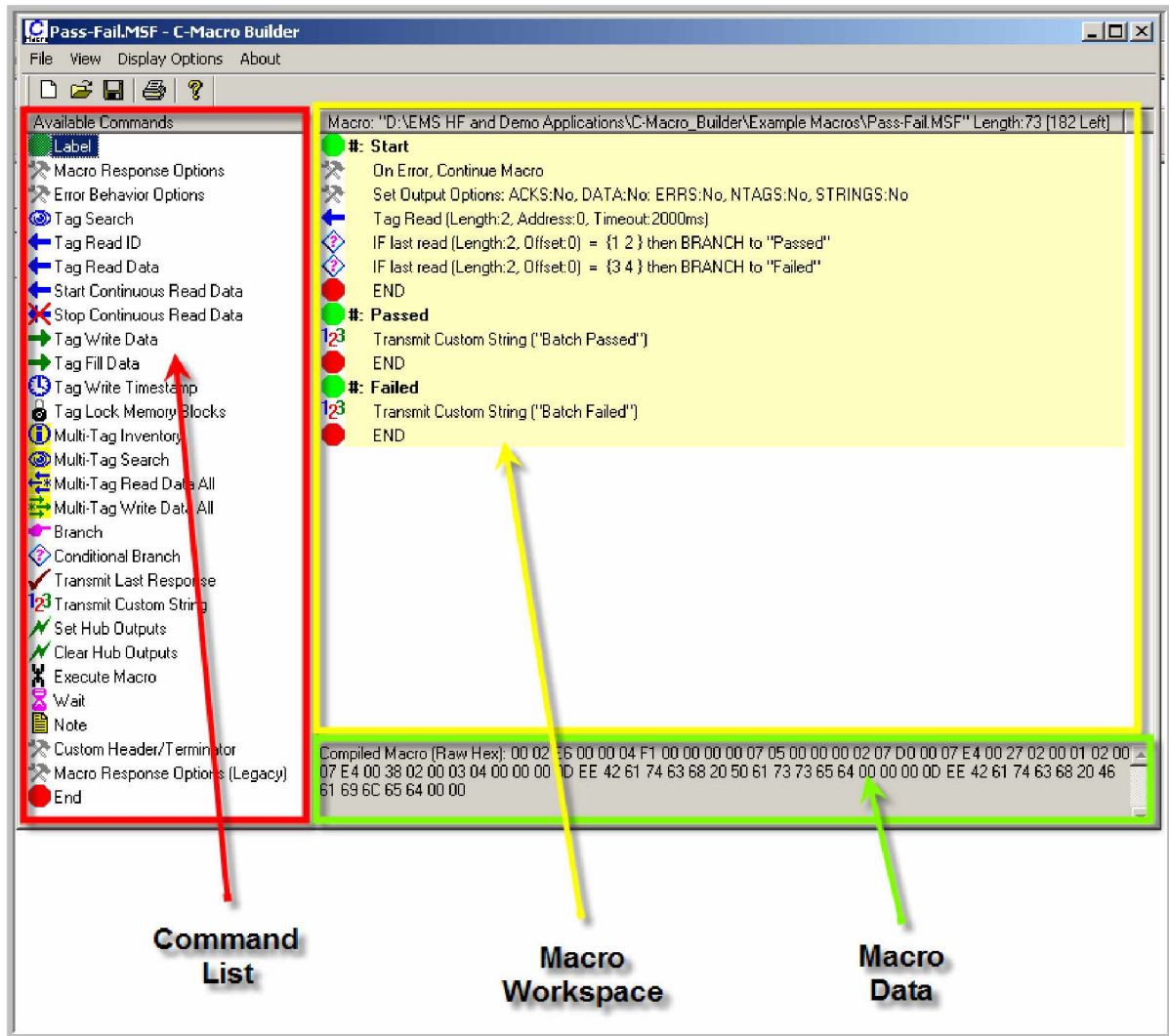


Figure 1-1: C-Macro Builder User Interface

- The pane on the left is the “*Command List*,” which displays an inventory of available command tasks for use in building a macro. Each Command List item will be addressed individually later in this guide.
- The large pane on the right is the “*Macro Workspace*.” This is the area where the macro is built. Each line represents one command task.
- Below the Macro Workspace (in the lower right corner) is the small “*Macro Data*” box, which displays the information in hex format, as it would appear after the macro is compiled.

### 1.1.1 Building Macros

To build a macro, command tasks are dragged, one at a time, from the Command List and dropped onto the desired location in the *Macro Workspace*.

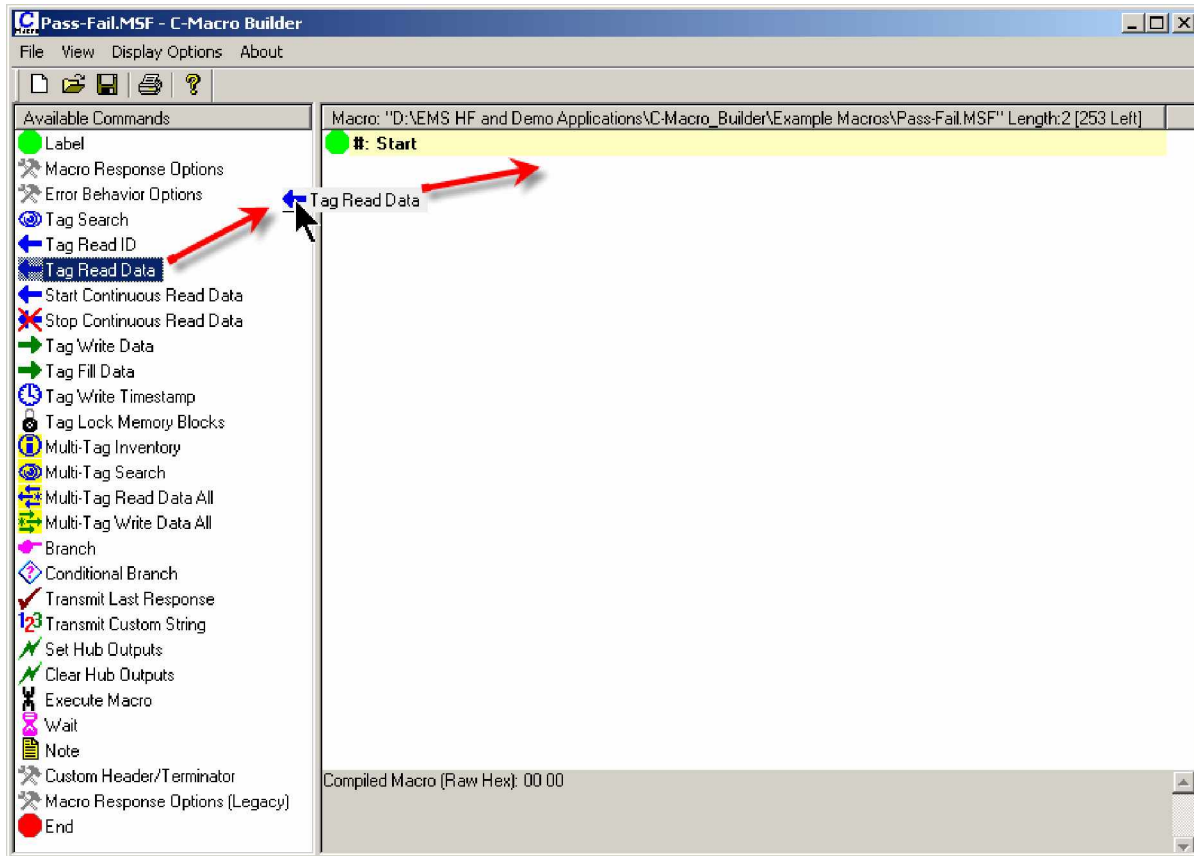
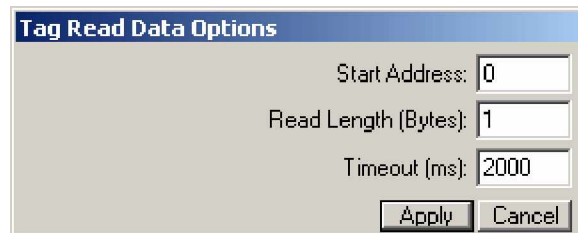


Figure 1-2: Dragging a Tag Read Data Command onto the Macro Workspace

When a command task is dragged and dropped onto the *Macro Workspace*, a popup options dialog box will appear, allowing the user to enter specific values for that command task.

For example, if you drop a “*Tag Read Data*” command task onto the workspace, the “*Tag Read Data Options*” dialog box will be displayed.



You then specify values in the fields of the dialog box – in this case, the *Start Address*, *Read Length* and *Timeout* values for the *Tag Read Data* command task.

Note that certain command tasks, such as “*Stop Continuous Read Data*” and “*Transmit Last Response*,” for example, have no editable parameters and, therefore, do not have associated popup options dialog boxes.

Once you have entered your values, click “*Apply*” to close the command task options dialog box and insert the command task into the *Macro Workspace*.

After a command task has been inserted, the parameter values may be edited later by double clicking the line item. Individual items can also be repositioned within the Macro Workspace by selecting and dragging them with the mouse.

When a compiled macro program is executed by an RFID controller, macro execution begins at the top, and each command task is executed in sequence. During execution, the controller will send a “response” back to the host after each command task is completed (unless you specify that a response not be generated by the controller).

There are also various “branching” command tasks that will be explained later that can be used to redirect the execution of commands to other parts of the macro.

While building a macro, the total length of the compiled string is displayed in the title bar above the Macro Workspace area. This status information allows you to gauge how much physical space remains for the particular macro, up to a maximum of 255 bytes.

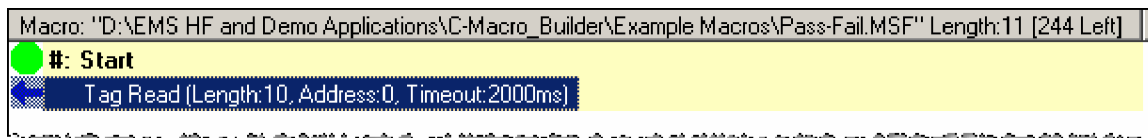


Figure 1-3: Macro Workspace - Status Information

### 1.1.2 Saving Macros

When saving a macro, C-Macro Builder “compiles” the list of command tasks into a string of byte values as the macro is being created. This string of bytes can be seen below the Macro Workspace, in the Macro Data box. This display area can be enabled or disabled from the “Display Options” menu.

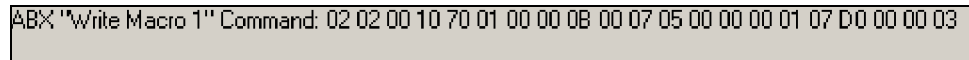
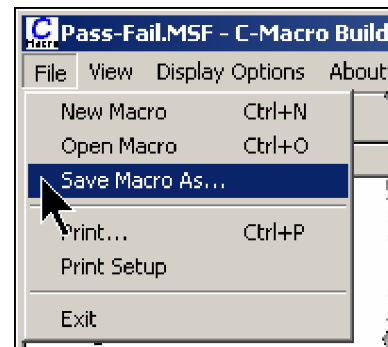


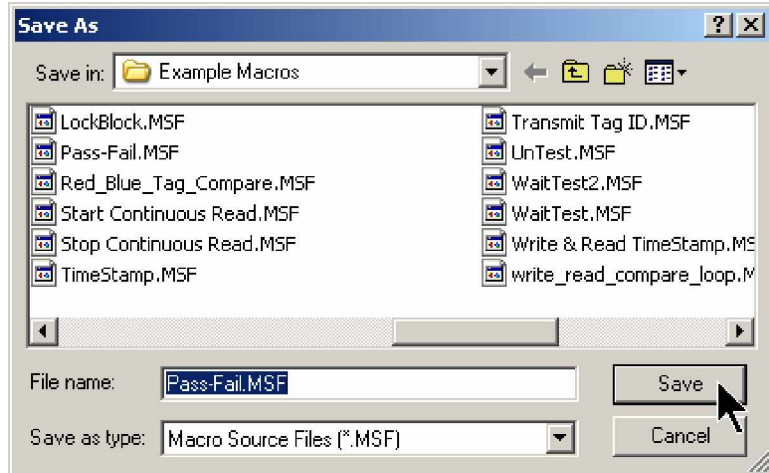
Figure 1-4: The Macro Data Box

To save the source file of a macro (the file that C-Macro Builder uses to create the “compiled” macro), click “Save Macro As” in the File menu.

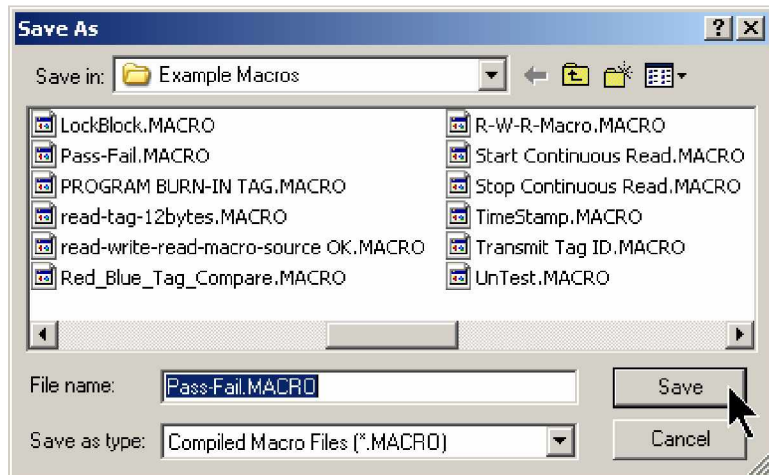




A “Save As” dialog box will be presented in which you can name the macro source file. “Macro Source Files” are saved with the file extension “.MSF.”



Immediately after saving the macro source file (\*.MSF), C-Macro Builder will prompt you to name and save the compiled macro file. Compiled macro files are saved with the file extension “.MACRO.”



## 1.2 LOADING MACROS INTO AN RFID CONTROLLER

The C-Macro Builder utility itself only builds and compiles macros. Loading macros into an EMS RFID controller is accomplished through the use of either the “*Cobalt HF TCP/IP Dashboard*” or the “*Cobalt HF Serial Dashboard*” (depending on the type of communications interface your application is incorporating).

Both versions of the Dashboard are available online at [www.ems-rfid.com](http://www.ems-rfid.com). Refer to the documentation included with the Dashboard for instructions on loading macros into supported EMS RFID controllers.

# CHAPTER 2: C-MACRO BUILDER COMMANDS

Below is the list of command tasks that can be individually inserted into RFID command macros using C-Macro Builder.

## 2.1 COMMAND LIST ICONS

- [Label](#)
- [Macro Response Options](#)
- [Error Behavior Options](#)
- [Tag Search](#)
- [Tag Read ID](#)
- [Tag Read Data](#)
- [Start Continuous Read Data](#)
- [Stop Continuous Read Data](#)
- [Tag Write Data](#)
- [Tag Fill Data](#)
- [Tag Write Timestamp](#)
- [Tag Lock Memory Blocks](#)
- [Multi-Tag Inventory](#)
- [Multi-Tag Search](#)
- [Multi-Tag Read Data All](#)
- [Multi-Tag Write Data All](#)
- [Branch](#)
- [Conditional Branch](#)
- [Transmit Last Response](#)
- [Transmit Custom String](#)
- [Set Hub Outputs](#)
- [Clear Hub Outputs](#)
- [Execute Macro](#)
- [Wait](#)
- [Note](#)
- [Custom Header/Terminator](#)
- [Macro Response Options \(Legacy\)](#)
- [End](#)



### 2.1.1 Label



**Labels** are used to identify specific entry points within a macro. Every macro has an initial Label called “**Start**” that is fixed at the beginning of the macro. This is the only Command List item that cannot be deleted from the Macro Workspace (although it can be renamed, if necessary).

Labels can be used to identify locations within a macro to which execution will branch (or “go to”) when a “*Branch*” instruction is encountered, (“branching” is described later in this chapter). Labels do not take up any “programming space” and technically do nothing by themselves. Users may add as many as they want.

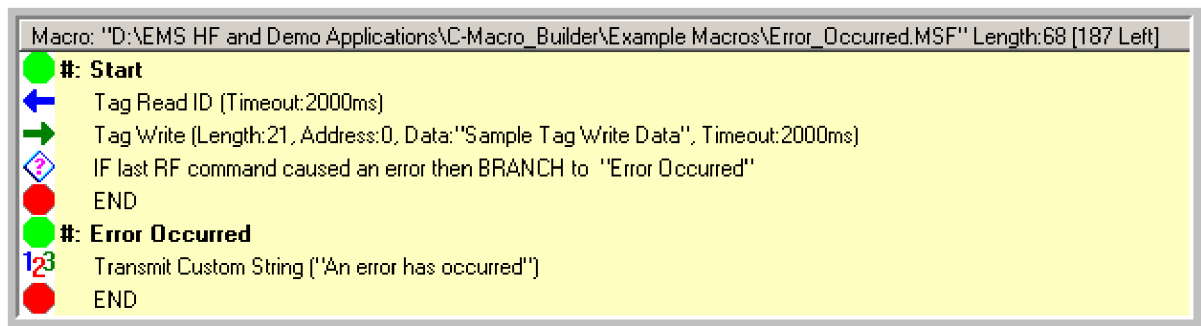
When a Label is inserted into a macro, the *Label Options* dialog box is displayed. You may enter a descriptive name in this box.



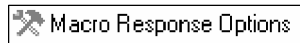
**NOTE:** Label names are **CASE SENSITIVE** and can contain up to 255 characters. In practice, however, Label names should be kept short (less than 32 characters long), so they can be easily read in the Macro Workspace. Label names can contain any printable characters, including spaces.

**EXAMPLE:**

In the example below, the controller will read the tag ID and then write “*Sample Tag Write Data*” to the tag beginning at tag address zero. If the write operation completes successfully, the macro will end. If an error occurs, the macro will branch to the Label “*Error Occurred*” at which time the custom string message “*An error has occurred*” will be delivered to the host.



## 2.1.2 Macro Response Options

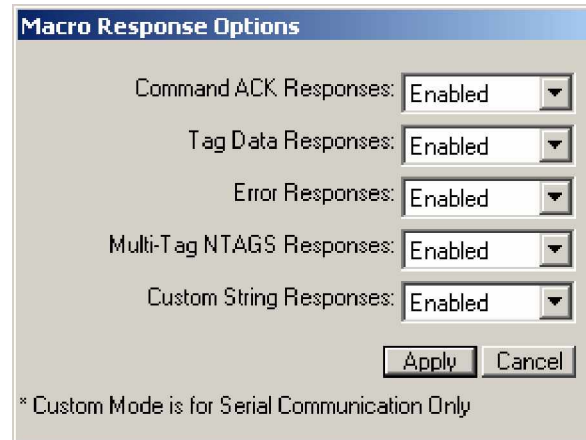


This command task allows the user to control how, when and if macro responses are returned to the host.

When this item is inserted into a macro, the *Macro Response Options* dialog box is displayed.

There are five basic types of macro responses (see below for descriptions):

- Command Acknowledgement Responses
- Tag Data Responses
- Error Responses
- Multi-Tag Number of Tags Responses
- Custom String Responses



Each of the response types has a drop-down menu for selecting a response status. The choices are **Disabled**, **Enabled** or **Custom**. By default, all macro response types are enabled, therefore all macro responses generated by RFID commands are sent back to the host.

**IMPORTANT NOTE:** Each drop-down menu's *Custom* option can be used in conjunction with the *Custom Header/Terminator* command task (see [Section 2.1.26 – Custom Header/Terminator](#)) to build macros that instruct the controller to return data from a tag **without** delivering command protocol overhead characters (such as the response header and terminator bytes).

It is for this reason that the *Custom* drop-down options should **NOT** be used with Subnet16 controllers (-485 models) or if you are planning to use the Cobalt HF Dashboard for anything other than loading and unloading macros. *Custom* drop-down options should only to be used with serial-based RFID controllers (and are not supported by HF-0405-Series controllers).

### MACRO RESPONSE DESCRIPTIONS

- **COMMAND ACK RESPONSES:** Enabling this option instructs the controller to return **all** command acknowledgement responses (“*command echoes*”) to the host.
- **TAG DATA RESPONSES:** Enabling this option instructs the controller to return **all** tag data responses to the host.
- **ERROR RESPONSES:** Enabling this option instructs the controller to return **all** error responses to the host.
- **MULTI-TAG NTAGS RESPONSES:** Enabling this option instructs the controller to return the number of tags read/written during multi-tag operations.
- **CUSTOM STRING RESPONSES:** Enabling this option instructs the controller to return a user-defined custom string response to the host when a *Transmit Custom String* command task is inserted into a macro (see [Section 2.1.20 – Transmit Custom String](#)).

**EXAMPLE 1:**

In the following example, the command task “*Macro Response Options*” is inserted into the macro and all response options are enabled. The controller will perform four different command tasks and will return four separate response messages to the host.

```
Macro: "D:\NEMS HF and Demo Applications\C-Macro_Builder\Example Macros\All_Macro_Responses.MSF" Length:35 [220 Left]
# Start
// In this macro, all responses are enabled.
Set Macro Response Options: ACKS:Yes, DATA:Yes, ERRS:Yes, NTAGS:Yes, STRINGS:Yes
Tag Search (Timeout:2000ms)
Tag Read ID (Timeout:2000ms)
Tag Read (Length:10, Address:0, Timeout:2000ms)
Tag Write Timestamp (Format:0, Address:11, Timeout:2000ms)
END
```

**EXAMPLE 2:**

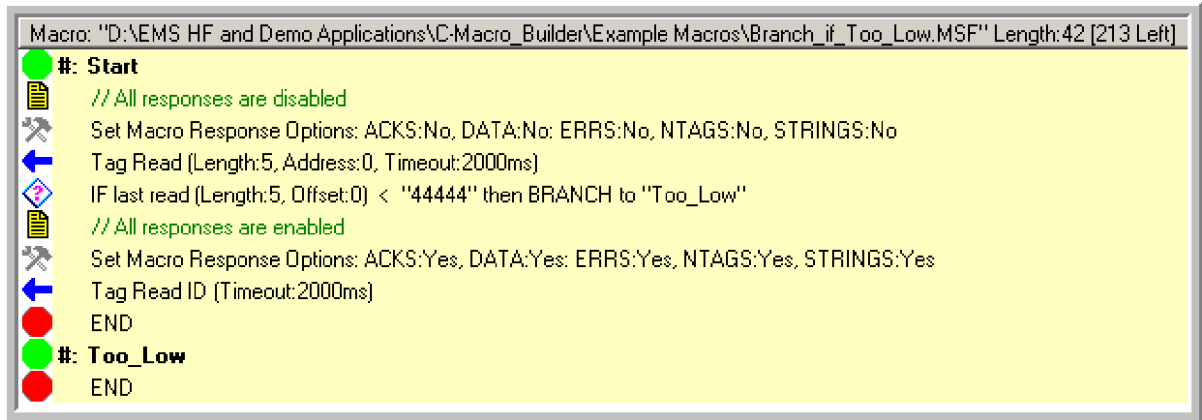
In the following example, the command task “*Macro Response Options*” is inserted into the macro and all response options are disabled. It is used to hide all macro responses (including “*Tag Not Found*” errors). If a tag ID is successfully read, it will be returned to the host (via the *Transmit Last Response* command task). If no tag is present, or if the tag ID is not successfully read, the macro will branch to the Label “*On\_Error*” and will then end.

```
Macro: "D:\NEMS HF and Demo Applications\C-Macro_Builder\Example Macros\Transmit Tag ID.MSF" Length:28 [227 Left]
# Start
// In this macro, all responses are disabled
// Only successfully read tag IDs will be returned
// And any "Tag Not Found" errors will be "hidden."
Set Macro Response Options: ACKS:No, DATA:No, ERRS:No, NTAGS:No, STRINGS:No
On Error, Continue Macro
Tag Read ID (Timeout:2000ms)
IF last RF command caused a "Tag Not Found" error then BRANCH to "On_Error"
Transmit Last Response
END
# On_Error
END
```

**EXAMPLE 3:**

In the following example, this command task is used twice. The first instance silences all macro responses before the first 5 bytes of a tag are read. If those 5 bytes are less than 44444, then the macro branches to the Label: *Too\_Low* and then ends.

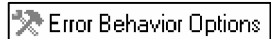
If the five bytes are greater than 44444, then the *Set Output Options* command task is executed (which re-enables all macro responses) and the tag ID is read and returned to the host.



```
Macro: "D:\EMS HF and Demo Applications\C-Macro_Builder\Example Macros\Branch_if_Too_Low.MSF" Length:42 [213 Left]
# Start
// All responses are disabled
Set Macro Response Options: ACKS:No, DATA:No, ERRS:No, NTAGS:No, STRINGS:No
Tag Read (Length:5, Address:0, Timeout:2000ms)
IF last read (Length:5, Offset:0) < "44444" then BRANCH to "Too_Low"
// All responses are enabled
Set Macro Response Options: ACKS:Yes, DATA:Yes, ERRS:Yes, NTAGS:Yes, STRINGS:Yes
Tag Read ID (Timeout:2000ms)
END
# Too_Low
END
```

This command task may be inserted anywhere in a macro, and when encountered, places the macro into the specified macro response mode from that point on.

### 2.1.3 Error Behavior Options



This command task allows the user choose whether to force the macro to “*halt on error*” or allow it to “*continue on error.*”

When this item is inserted into a macro, the *Error Behavior Options* dialog box is displayed.

By default, when a macro begins, all errors generated during execution cause the macro to stop at that point. This includes “*tag not found*” errors.



#### On Error, Halt Macro

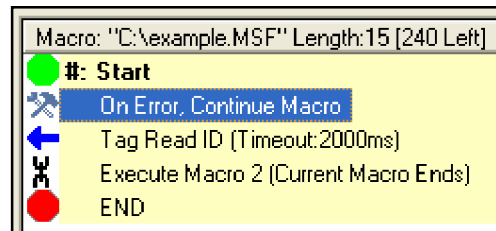
When an error is encountered with this command task option selected, the macro will stop execution immediately when an error occurs.

#### On Error, Continue Macro

When an error is encountered and this command task option is selected, the macro will not stop execution after an error and will continue on to the next instruction.

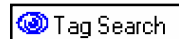
#### EXAMPLE:

This command task is used to allow the macro to continue even if the Tag Read ID command task fails. Without inserting this command task, a “*Tag Not Found*” error would cause the macro to immediately end, and the “*Execute Macro 2*” command task would not be performed.





## 2.1.4 Tag Search



This command task instructs the RFID controller to search for the presence of a tag in its RF field, without performing an actual read or write operation. The resulting response, either a *'tag found'* message or a *'tag not found'* error, is sent to the host.

When the item is inserted into a macro, the *Tag Search Options* dialog box is displayed.



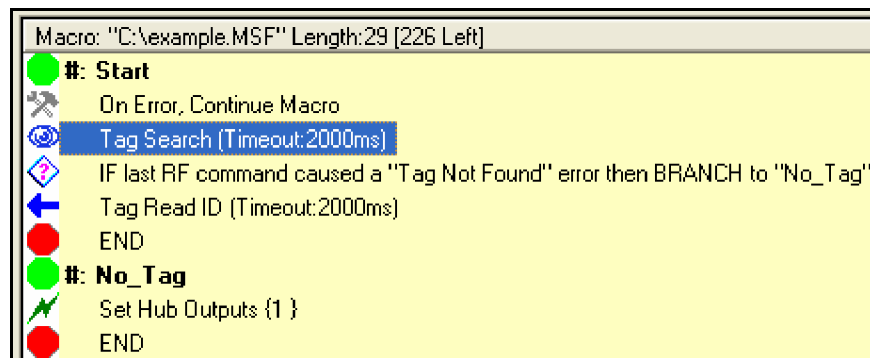
The only editable option is the search *"Timeout"* value (measured in milliseconds).

### EXAMPLE:

In the following example, the *"Tag Search"* command task is inserted into a macro to determine if a tag is present.

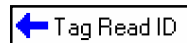
If the *Timeout* expires and no tag has been found, the macro branches to the Label: *No\_Tag*. Hub Output 1 will be set and a *"tag not found"* error will be sent to the host.

If a tag is found, the tag ID will be returned along with a *"tag found"* message to the host.



There are only rare cases where this command task would actually be a better choice than a normal *Tag Read* or *Tag Write* – as those command tasks also return the same *"tag not found"* error if a tag is not present. However, this command task can be used to save macro space because it requires fewer bytes than the *Tag Read* or *Tag Write* command tasks.

### 2.1.5 Tag Read ID



Tag Read ID

This command task instructs the controller to retrieve a tag's unique identification number (tag ID). If the tag ID cannot be read, the appropriate error is sent to the host.

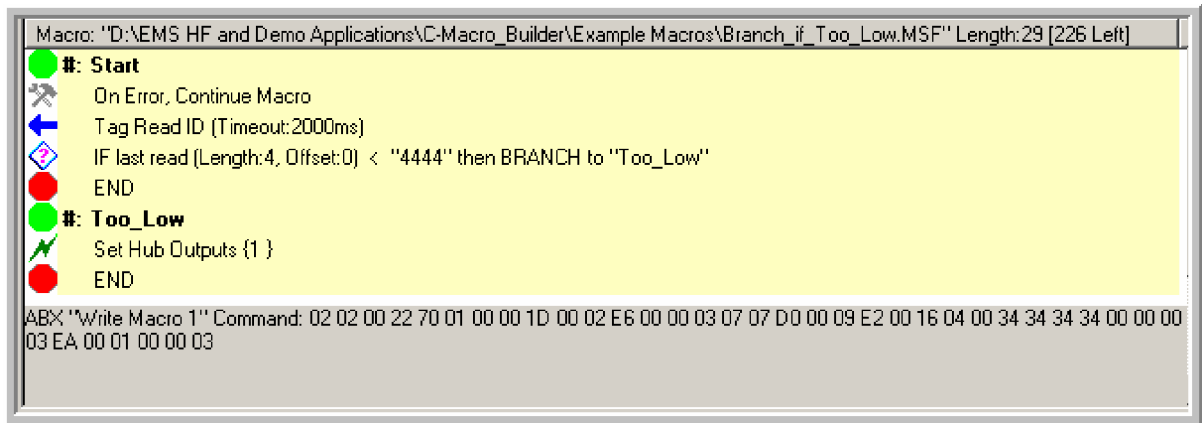
When the item is inserted into a macro, the *Tag Read ID Options* dialog box is displayed.



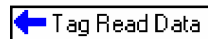
The only editable option is the "Timeout" value (measured in milliseconds).

**EXAMPLE:**

In the following example, the "Tag Read ID" command task is used to instruct the controller to retrieve the tag ID. If the first four bytes of the tag ID are less than "4444" then Hub output 1 is set, otherwise, the macro ends.



## 2.1.6 Tag Read Data



This command task performs a “Tag Read Data” command. When this command task is executed, the specified tag data is read and is sent to the host. If the tag data is not read, the appropriate error is sent to the host.

When the item is inserted into a macro, the *Tag Read Data Options* dialog box is displayed.

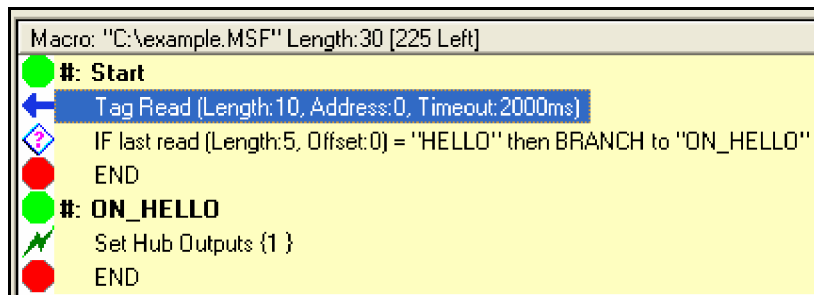
The editable options for this command task are:

The dialog box titled "Tag Read Data Options" contains three input fields: "Start Address" with the value 0, "Read Length (Bytes)" with the value 10, and "Timeout (ms)" with the value 2000. At the bottom right, there are "Apply" and "Cancel" buttons.

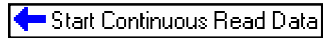
- **Start Address:** (0 to 65535, depending on tag length)
- **Read Length (Bytes):** (1 to 247, depending on tag length)
- **Timeout (ms):** (0 to 65535)

### EXAMPLE:

In the following example, the “*Tag Read Data*” command task is used to read 10 bytes from a tag starting at address 0, and the result is transmitted back to the host. If the first 5 bytes read “*HELLO*,” then Hub output 1 is set, otherwise, the macro ends.

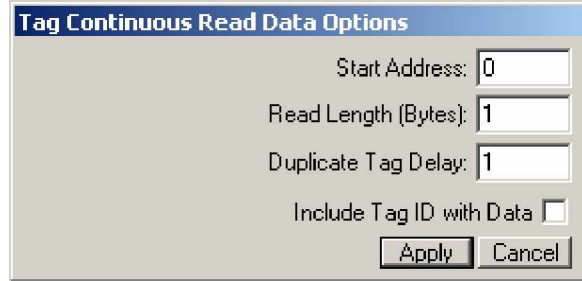


## 2.1.7 Start Continuous Read Data

 This command task instructs the controller to enter “Continuous Read” mode.

When this command task is executed, the controller will begin operating in “Continuous Read Mode” after the macro is finished.

This command task is not used to read actual data for processing within a macro - it only instructs the controller to begin continuous reads once the macro terminates.



The dialog box titled "Tag Continuous Read Data Options" contains the following fields and controls:

- Start Address: 0
- Read Length (Bytes): 1
- Duplicate Tag Delay: 1
- Include Tag ID with Data:
- Buttons: Apply, Cancel

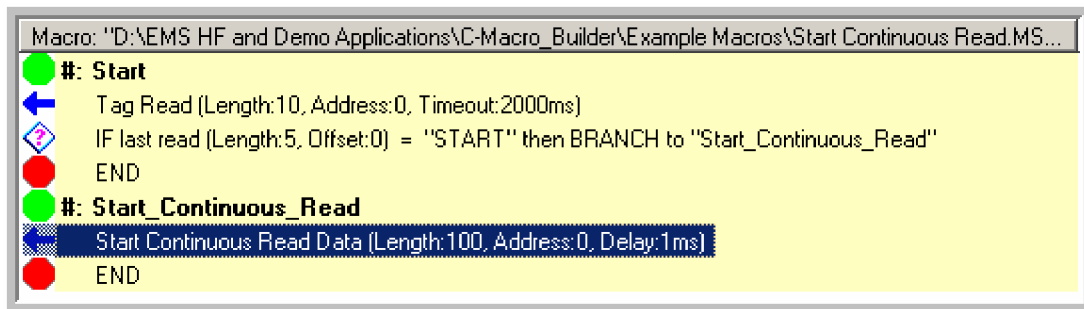
When the item is inserted into a macro, the *Tag Continuous Read Data Options* dialog box is displayed:

The editable options for this command task are

- **Start Address:** (0 to 65535, depending on tag length)
- **Read Length (Bytes):** (0 to 247, depending on tag length)
- **Duplicate Read Delay:** (0 to 255 – measured in seconds)
- **Include Tag ID with Data** (placing a check in this box will instruct the controller to retrieve the tag ID number in front of any read data)

### EXAMPLE:

In the following example, 10 bytes are read starting at tag address 0. If the first five bytes equal “**START**”, then the macro branches to the Label “*Start\_Continuous\_Read*,” which places the controller into “*Continuous Read*” mode where it will read 100 bytes from tag address 0. The continuous read operation will begin after the macro terminates.

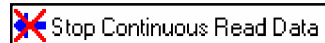


Macro: "D:\EMS HF and Demo Applications\C-Macro\_Builder\Example Macros\Start Continuous Read.MS..."

```

# Start
Tag Read (Length:10, Address:0, Timeout:2000ms)
IF last read (Length:5, Offset:0) = "START" then BRANCH to "Start_Continuous_Read"
END
# Start_Continuous_Read
Start Continuous Read Data (Length:100, Address:0, Delay:1ms)
END
    
```

## 2.1.8 Stop Continuous Read Data



Stop Continuous Read Data

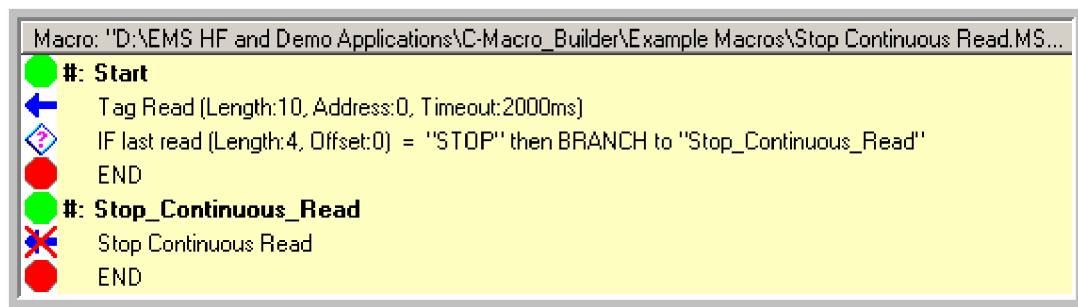
This command task instructs the controller to stop continuous reading. It is typically used to halt a continuous read operation when a controller is configured to enter continuous read mode upon power-up or reset. Note that inserting a *Stop Continuous Read* command task into a macro that already has a *Start Continuous Read* command task will cancel out the continuous read operation, as a macro must conclude before the initial continuous read operation can take place.

When this command task is executed, the controller will stop operating in “*Continuous Read Mode*.” When the macro terminates, the controller will not resume continuous reading. Always stop continuous read operations before executing another tag command task.

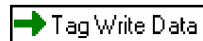
There are no editable items for this command task, therefore, when the item is inserted into a macro, no dialog box is displayed.

### EXAMPLE:

In the following example, 10 bytes are read from tag address 0. If the first four bytes equal “**STOP**”, then the macro branches to the Label “*Stop\_Continuous\_Read*” and the controller is instructed to stop “*Continuous Read Mode*.”



### 2.1.9 Tag Write Data



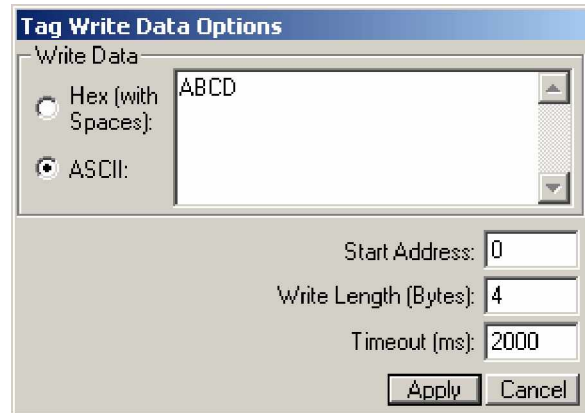
This command task instructs the controller to perform a “*Tag Write Data*” command.

When this command task is executed and the specified data is written, a “*Write Succeeded*” response is sent to the host. If the tag write fails, the appropriate error message is sent to the host.

When the item is inserted into a macro, the *Tag Write Data Options* dialog box is displayed:

The editable options for this command task are:

- **Write Data:** (0 to 241 bytes of either HEX or ASCII data, depending on tag length)
- **Start Address:** (0 to 65535, depending on tag length)
- **Write Length (Bytes):** (0 to 241, this field will be automatically populated based on the number of bytes entered in the *Write Data* box)
- **Timeout (ms):** (0 to 65535)

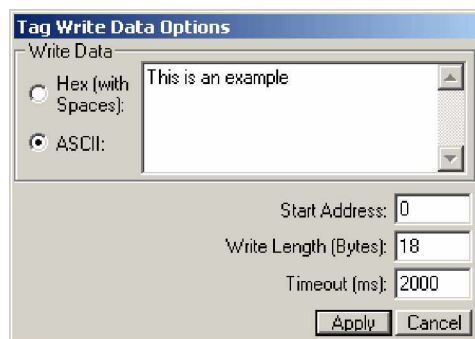


#### WRITE DATA BOX

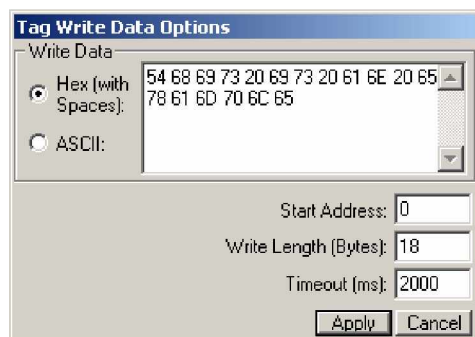
By selecting the appropriate radio button, the display mode of this command task can be set to either HEX (with Spaces) or ASCII.

Enter data in the “*Write Data*” box in either two digit HEX byte values separated by spaces or in ASCII text.

#### ASCII DATA EXAMPLE:



#### HEX DATA EXAMPLE:

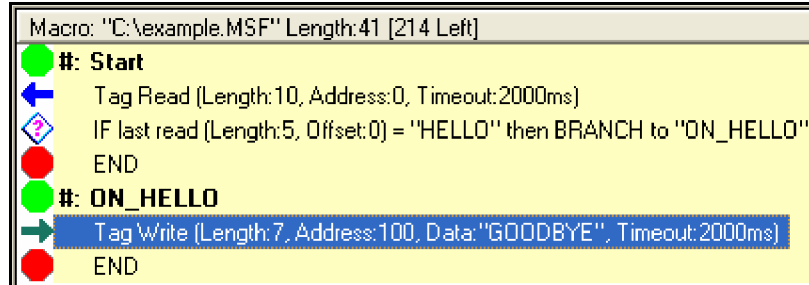


Note that the two examples contain identical data; however, the actual data is always written to a tag in Hex format.

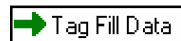
**NOTE:** Entering the ASCII string "12345" actually writes to a tag the HEX values 31 32 33 34 35. It is important to realize that when entering numbers in ASCII mode, that these are the ASCII characters themselves, and not the numerical equivalent. If you want to write the actual byte values 01 02 03 04 05, you would need to use HEX data entry mode.

**EXAMPLE:**

In the following example, the first 10 bytes of a tag are read and sent to the host. If the first 5 bytes equal "**HELLO**", then the "*Tag Write Data*" command task is executed to write the string "**GOODBYE**" to the tag at address 100.



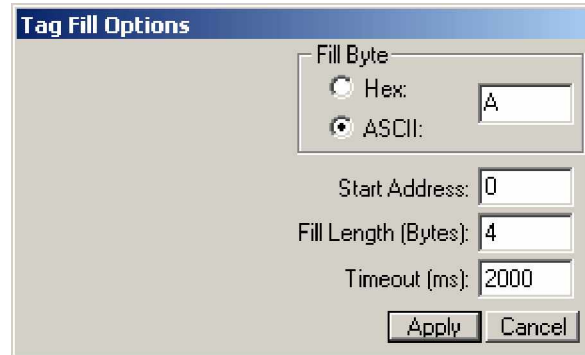
### 2.1.10 Tag Fill Data



This command task instructs the controller to perform a “*Tag Fill Data*” command.

When this command task is executed and the specified *Fill Byte* is successfully written across the specified range in the tag, a “*Tag Fill Succeeded*” message is sent to the host. If the Tag Fill operation fails, the appropriate error message is sent to the host.

When the item is inserted into a macro, the *Tag Fill Options* dialog box is displayed.



The editable options for this command task are

- **Fill Byte:** (1 byte either ASCII or HEX data)
- **Start Address:** (0 to 65535, depending on tag length)
- **Fill Length (Bytes):** (0 to 65535, depending on tag length, 0 = entire tag)
- **Timeout (ms):** (0 to 65535)

By selecting the appropriate radio button, the display mode of this command task can be set to either HEX or ASCII.

Enter the *Fill Byte* either as a 2-digit HEX byte value or as a single ASCII character.

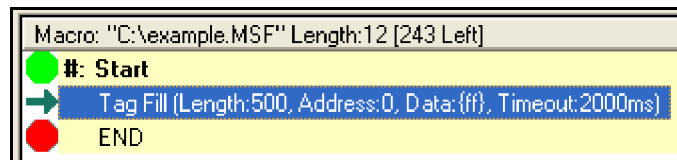
The *Fill Data* written to the tag is the same either way – this is just for convenience of data entry, or for human-readability.

**NOTE:** Entering the ASCII character “1” actually writes the HEX value “31” to the tag.

It is important to realize that when entering numbers in ASCII mode, that these are the ASCII characters themselves, and not the numerical equivalent. Therefore, if you want to write the value 01, you would need to use HEX data entry mode.

**EXAMPLE:**

In the following example, the “*Tag Fill Data*” command task is used to write the byte value *0xFF* from address 0 to address 500 of the tag.





### 2.1.11 Tag Write Timestamp

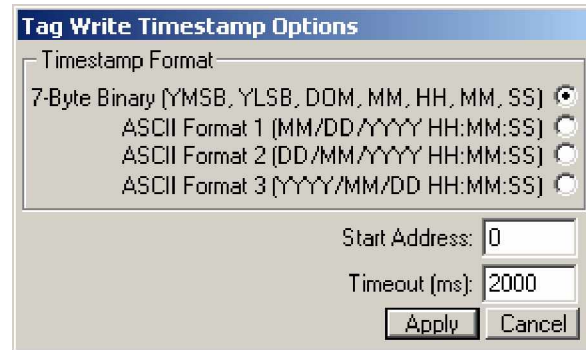


This command task instructs the controller to write the date and time, in the specified timestamp format, to a tag.

When the item is inserted into a macro, the *Tag Write Timestamp Options* dialog box is displayed, which allows the user to select their preferred timestamp format.

Choices for *Timestamp Format* are:

- **Seven-Byte Binary**  
(Year MSB, Year LSB, Day of Month, Month, Hour, Minute, Second)
- **ASCII Format 1**  
(Month/Day/Year  
Hour:Minute:Second)
- **ASCII Format 2**  
(Day/Month/Year  
Hour:Minute:Second)
- **ASCII Format 3**  
(Year/Month/Day  
Hour:Minute:Second)



The editable options for this command task are

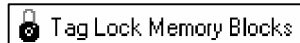
- **Start Address:** (0 to 65535, depending on tag length)
- **Timeout (ms):** (0 to 65535)

The *Start Address* identifies where the timestamp will be written on the tag.

#### TIMESTAMP NOTES:

- To set the date and time of the controller to that of your host computer, execute **CBx Command 0x4E**, **ABx Fast Command 0x51** or use either version of the **Cobalt HF Dashboard** software utility.
- Anytime power is cycled to an RFID controller, the date and time will need to be reset, as these units do not contain a battery-backed-up clock.
- HF-0405-Series controllers use a software-based clock. Timing accuracy is dependent on the controller's microprocessor activity and not on a hardware-implemented real time clock. All other Cobalt-series RFID controllers have hardware-implement real time clocks.

## 2.1.12 Tag Lock Memory Blocks

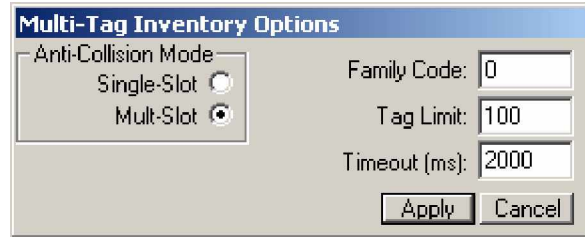


This command task allows the user to write protect or lock a contiguous segment of tag memory from being

overwritten.

This command supports **ISO 15693** compliant RFID tags only.

Depending on the architecture of the tag used, a block of tag memory can be either 4-bytes or 8-bytes. The memory in EMS LRP-Series (ISO 15693 compliant) RFID tags is arranged in 4-byte blocks. EMS T-Series (ISO 15693 compliant) RFID tags have 8-byte memory blocks.



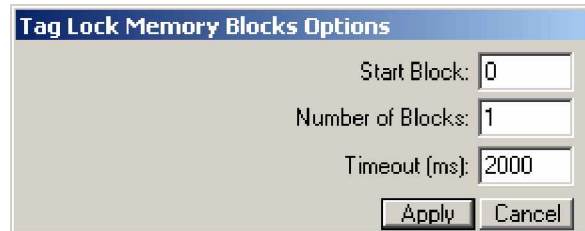
Users should be aware of the memory architecture and block size of their tag before using this command.

**IMPORTANT NOTE:** Extreme caution should be taken when using this command. Once a block of tag memory is locked, it cannot be unlocked and all data written to the block is permanent.

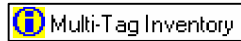
When the item is inserted into a macro, the *Lock Memory Blocks Options* dialog box is displayed.

The parameters for this command task are:

- **Start Block:** (the location of the block where lock will begin)
- **Number of Blocks:** (the number of blocks that will be locked)
- **Timeout (ms):** (the length of time set for the duration of the operation)



### 2.1.13 Multi-Tag Inventory



This command task instructs the controller to search its RF field for the presence of RFID tags and retrieve the tag ID number of all tags identified.

This command supports **ISO 15693** compliant RFID tags only.

When the item is inserted into a macro, the *Multi-Tag Inventory Options* dialog box is displayed.

The parameters for this command task are:

#### 2.1.13.1 Anti-Collision Mode

The **Anti-collision Mode** parameter controls the tag-reading algorithm used to achieve the fastest reading speed for the number of tags expected in RF range at any given moment. This parameter helps the controller avoid data collisions when simultaneously reading multiple tags.

The choices for the *Anti-collision Mode* parameter are **Single-Slot** or **Multi-Slot**.

- **Single-Slot:** Setting this parameter to *Single-Slot* utilizes a single time slot under which the requested data from all tags is transferred to the controller as soon as it becomes available. This setting can result in faster tag read performance when only a few tags are expected in the RF field.
- **Multi-Slot:** Setting this parameter to *Multi-Slot* implements a system of 16 time slots. To avoid data collisions when the controller encounters multiple tags simultaneously, data requested from each tag is transferred to the controller only during the time slot that matches a specific pattern in the tag ID number.

#### 2.1.13.2 Family Code (AFI)

The **Family Code** parameter (*0x00 – 0xFF*) can be used in multi-tag commands to specify a subset of tags when many are identified simultaneously in RF range. The parameter allows the user to filter tags based on a pre-written value stored at a special tag location tag.

For example, if the *Family Code* value is set to one (*0x01*), only those tags with the pre-written *Family Code* value of *0x01* will respond to the given command. When a *Family Code* value of zero (*0x00*) is set, all tags within RF range will respond to the command.

#### 2.1.13.3 Tag Limit

The **Tag Limit** parameter is used to indicate the highest number of tags expected simultaneously in RF range for the given multi-tag operation. By default, this value is set to **100**, which is also the maximum *Tag Limit* value. The actual number entered should be set in relation to the greatest number of tags that could possibly be present in the reading field at any one time.

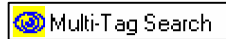
By lowering the default *Tag Limit* value, users can minimize the number of attempted read/write operations the controller will make per execution (users will not have to wait for the *Timeout* to expire). A low *Tag Limit* value can also speed up multi-tag operations when only a small group of tags will be present at any given moment.

Setting the proper *Tag Limit* value is therefore a tradeoff between the number of expected tags in the reading field, and the time required to read/write to them.

#### **2.1.13.4 Timeout (ms)**

The ***Timeout*** indicates the length of time set for the duration of the operation. The command will terminate when either the *Tag Limit* value or the *Timeout* value have been met.

### 2.1.14 Multi-Tag Search



This command task instructs the controller to search its RF field for the presence of RFID tags and retrieve the number of tags identified.

This command supports **ISO 15693** compliant RFID tags only.

When the item is inserted into a macro, the *Multi-Tag Search Options* dialog box is displayed.

The parameters for this command task are:

A screenshot of the 'Multi-Tag Search Options' dialog box. It has a title bar with the text 'Multi-Tag Search Options'. Below the title bar, there are two sections. The first section is 'Anti-Collision Mode' with two radio buttons: 'Single-Slot' (unselected) and 'Multi-Slot' (selected). The second section contains three input fields: 'Family Code:' with the value '0', 'Tag Limit:' with the value '100', and 'Timeout (ms):' with the value '2000'. At the bottom right of the dialog are two buttons: 'Apply' and 'Cancel'.

#### 2.1.14.1 Anti-Collision Mode

The **Anti-collision Mode** parameter controls the tag-reading algorithm used to achieve the fastest reading speed for the number of tags expected in RF range at any given moment. This parameter helps the controller avoid data collisions when simultaneously reading multiple tags.

The choices for the *Anti-collision Mode* parameter are **Single-Slot** or **Multi-Slot**.

- **Single-Slot:** Setting this parameter to *Single-Slot* utilizes a single time slot under which the requested data from all tags is transferred to the controller as soon as it becomes available. This setting can result in faster tag read performance when only a few tags are expected in the RF field.
- **Multi-Slot:** Setting this parameter to *Multi-Slot* implements a system of 16 time slots. To avoid data collisions when the controller encounters multiple tags simultaneously, data requested from each tag is transferred to the controller only during the time slot that matches a specific pattern in the tag ID number.

#### 2.1.14.2 Family Code (AFI)

The **Family Code** parameter can be used in multi-tag commands to specify a subset of tags when many are identified simultaneously in RF range. The parameter allows the user to filter tags based on a pre-written value stored at a special tag location tag.

For example, if the *Family Code* value is set to one (*0x01*), only those tags with the pre-written *Family Code* value of *0x01* will respond to the given command. When a *Family Code* value of zero (*0x00*) is set, all tags within RF range will respond to the command.

#### 2.1.14.3 Tag Limit

The **Tag Limit** parameter is used to indicate the highest number of tags expected simultaneously in RF range for the given multi-tag operation. By default, this value is set to **100**, which is also the maximum *Tag Limit* value. The actual number entered should be set in relation to the greatest number of tags that could possibly be present in the reading field at any one time.

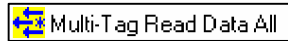
By lowering the default *Tag Limit* value, users can minimize the number of attempted read/write operations the controller will make per execution (users will not have to wait for the *Timeout* to expire). A low *Tag Limit* value can also speed up multi-tag operations when only a small group of tags will be present at any given moment.

Setting the proper *Tag Limit* value is therefore a tradeoff between the number of expected tags in the reading field, and the time required to read/write to them.

#### 2.1.14.4 Timeout (ms)

The ***Timeout*** indicates the length of time set for the duration of the operation. The command will terminate when either the *Tag Limit* value or the *Timeout* value have been met.

## 2.1.15 Multi-Tag Read Data All



Multi-Tag Read Data All

This command task instructs the controller to retrieve the specified number of bytes from each tag in its RF field.

This command supports **ISO 15693** compliant RFID tags only.

When the item is inserted into a macro, the *Multi-Tag Read Data All Options* dialog box is displayed.

The parameters for this command task are:

### 2.1.15.1 Anti-Collision Mode

The **Anti-collision Mode** parameter controls the tag-reading algorithm used to achieve the fastest reading speed for the number of tags expected in RF range at any given moment. This parameter helps the controller avoid data collisions when simultaneously reading multiple tags.

The choices for the *Anti-collision Mode* parameter are **Single-Slot** or **Multi-Slot**.

- **Single-Slot:** Setting this parameter to *single-slot* utilizes a single time slot under which the requested data from all tags is transferred to the controller as soon as it becomes available to the controller. This setting can result in faster tag read performance when only a few tags are expected in the RF field.
- **Multi-Slot:** Setting this parameter to *multi-slot* implements a system of 16 time slots. To avoid data collisions when the controller encounters multiple tags simultaneously, data requested from each tag is transferred to the controller only during the time slot that matches a specific pattern in the tag ID number.

### 2.1.15.2 Start Address

This parameter identifies the tag location where the read operation will begin (zero = begin reading at the first available byte of tag memory)

### 2.1.15.3 Read Length (Bytes)

This parameter represents the number of bytes that are to be read during the read operation.

### 2.1.15.4 Family Code (AFI)

The **Family Code** parameter can be used in multi-tag commands to specify a subset of tags when many are identified simultaneously in RF range. The parameter allows the user to filter tags based on a pre-written value stored at a special location on the tag.

For example, if the *Family Code* value is set to one (0x01), only those tags with the pre-written *Family Code* value of 0x01 will respond to the given command. When a *Family Code* value of zero (0x00) is set for this parameter, all tags within RF range will respond to the command.

#### 2.1.15.5 Tag Limit

The **Tag Limit** parameter is used to indicate the highest number of tags expected simultaneously in RF range for the given multi-tag operation. By default, this value is set to **100**, which is also the maximum **Tag Limit** value. The actual number entered should be set in relation to the greatest number of tags that could possibly be present in the reading field at any one time.

By lowering the default **Tag Limit** value, users can minimize the number of attempted read/write operations the controller will make per execution (users will not have to wait for the **Timeout** to expire). A low **Tag Limit** value can also speed up multi-tag operations when only a small group of tags will be present at any given moment.

Setting the proper **Tag Limit** value is therefore a tradeoff between the number of expected tags in the reading field, and the time required to read/write to them.

#### 2.1.15.6 Timeout (ms)

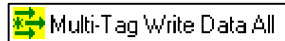
The **Timeout** indicates the length of time set for the duration of the operation. The command will terminate when either the **Tag Limit** value or the **Timeout Value** have been met.

#### 2.1.15.7 Include Tag ID with Data (Checkbox)

Place a check in this box, to have the controller retrieve the tag's ID number in front of the read data.



## 2.1.16 Multi-Tag Write Data All



This command task instructs the controller to write the specified data to all tags in its RF field.

This command supports **ISO 15693** compliant RFID tags only.

When the item is inserted into a macro, the *Multi-Tag Write Data All Options* dialog box is displayed.

The parameters for this command task are:

### 2.1.16.1 Write Data

Select whether the data to be written to each tag will be in HEX or ASCII format and then enter the data in the *Write Data* box.

### 2.1.16.2 Anti-Collision Mode

The **Anti-collision Mode** parameter controls the tag-reading algorithm used to achieve the fastest reading speed for the number of tags expected in RF range at any given moment. This parameter helps the controller avoid data collisions when simultaneously reading multiple tags.

The choices for the *Anti-collision Mode* parameter are **Single-Slot** or **Multi-Slot**.

- **Single-Slot:** Setting this parameter to *single-slot* utilizes a single time slot under which the requested data from all tags is transferred to the controller as soon as it becomes available. This setting can result in faster tag read performance when only a few tags are expected in the RF field.
- **Multi-Slot:** Setting this parameter to *multi-slot* implements a system of 16 time slots. To avoid data collisions when the controller encounters multiple tags simultaneously, data requested from each tag is transferred to the controller only during the time slot that matches a specific pattern in the tag ID number.

### 2.1.16.3 Start Address

This parameter identifies the tag location where the write operation will begin (zero = begin writing to the first available byte of tag memory)

### 2.1.16.4 Write Length (Bytes):

This parameter represents the number of bytes that are to be written during the operation. This field will be automatically populated by the application based on the number of bytes entered in the *Write Data* box.

### 2.1.16.5 Timeout (ms)

The **Timeout** indicates the length of time set for the duration of the operation. The command will terminate when either the *Tag Limit* value or the *Timeout* value have been met.

#### 2.1.16.6 Family Code (AFI)

The **Family Code** parameter can be used in multi-tag commands to specify a subset of tags when many are identified simultaneously in RF range. The parameter allows the user to filter tags based on a pre-written value stored at a special location on the tag.

For example, if the *Family Code* value is set to one (*0x01*), only those tags with the pre-written *Family Code* value of *0x01* will respond to the given command. When a *Family Code* value of zero (*0x00*) is set for this parameter, all tags within RF range will respond to the command.

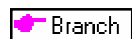
#### 2.1.16.7 Tag Limit

The **Tag Limit** parameter is used to indicate the highest number of tags expected simultaneously in RF range for the given multi-tag operation. By default, this value is set to **100**, which is also the maximum *Tag Limit* value. The actual number entered should be set in relation to the greatest number of tags that could possibly be present in the reading field at any one time.

By lowering the default *Tag Limit* value, users can minimize the number of attempted read/write operations the controller will make per execution (users will not have to wait for the *Timeout* to expire). A low *Tag Limit* value can also speed up multi-tag operations when only a small group of tags will be present at any given moment.

Setting the proper *Tag Limit* value is therefore a tradeoff between the number of expected tags in the reading field, and the time required to read/write to them.

### 2.1.17 Branch



Branch

This command task causes the flow of macro execution to unconditionally branch to another location in the macro. When using a “Branch” command task, you must first create a *Label* to identify the destination of the branch operation. This is the equivalent of a “Go To” command.

When the item is inserted into a macro, the *Branch Options* dialog box is displayed.

The only editable option for this command task is:



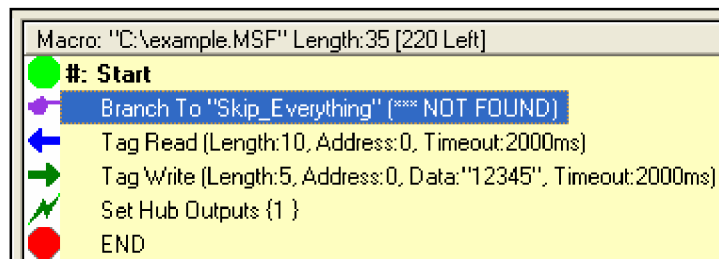
**Branch to Label Name:** Enter the name of the *Label* to which you want the macro to branch.

If the Branch to Label does not actually exist at the time you insert this command task, a warning message will be displayed next to the command item in the Macro Workspace that reads “(\*\* **NOT FOUND**).” This is an indication that C-Macro Builder does not yet know where you want the macro to branch. You will still need to insert a *Label* with the appropriate name.

**NOTE:** With all branch command tasks, care should be taken not to cause macros to enter “infinite loops,” unless that is the desired behavior.

#### EXAMPLE 1:

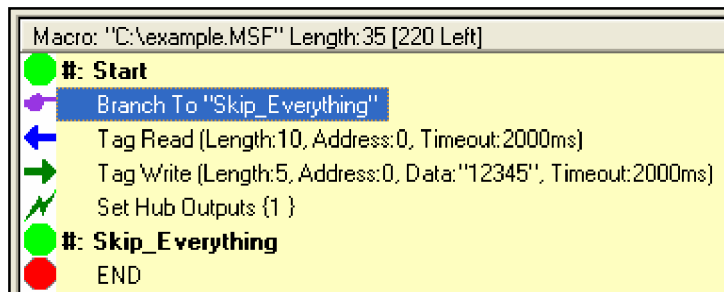
In this example, a branch to the label named “*Skip Everything*” has been inserted at the beginning of the macro. Because there is no label named “*Skip Everything*”, the warning message (\*\* **NOT FOUND**) is displayed.



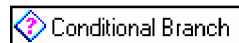
**NOTE:** If the macro in example 1 were to be loaded, and executed, an error would occur.

#### EXAMPLE 2:

In this example, the appropriate label called “*Skip Everything*” has been inserted before the end of the macro. When this macro is executed, the branch instruction will cause the macro to skip over the next three command tasks and resume executing at the “*Skip Everything*” label (which, in this case, ends the macro).



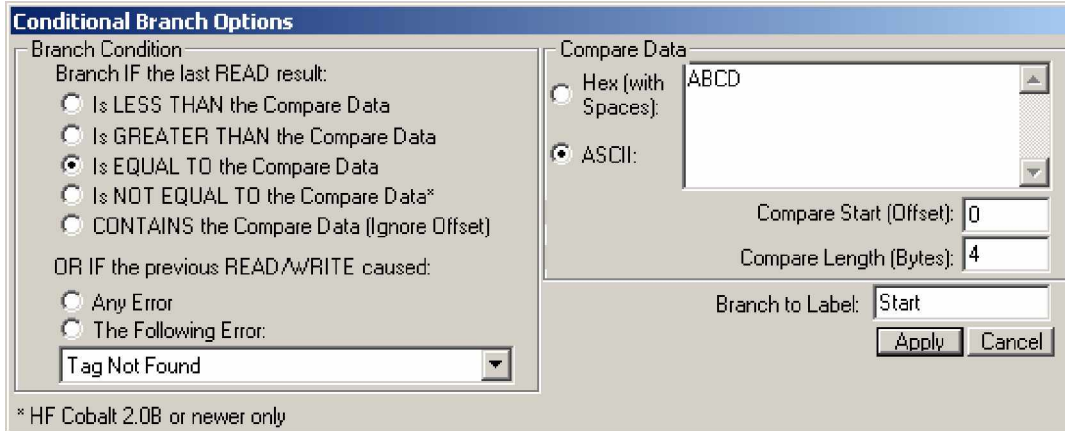
## 2.1.18 Conditional Branch



This command task causes the macro execution flow to conditionally branch to another location in the macro, based on the results of data comparisons or errors from a previous RFID command task (*Tag Search*, *Tag Read ID*, *Tag Read Data*, *Tag Write Data*, or *Tag Fill Data*).

The *Conditional Branch* command task requires that a *Label* exists at the destination of the branch. This is the equivalent of an “If X occurs, then go to Y” command.

When the item is inserted into a macro, the *Conditional Branch Options* dialog box is displayed.



### 2.1.18.1 Branch Conditions

On the left of the *Conditional Branch Options* box, are seven types of branch conditions.

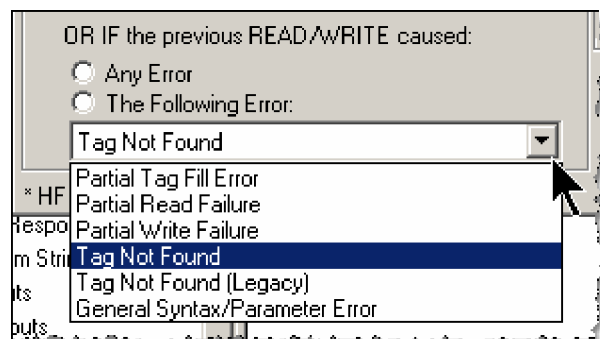
#### **Branch IF the last READ result:**

- Is LESS THAN the Compare Data
- Is GREATER THAN the Compare Data
- Is EQUAL TO the Compare Data
- Is NOT Equal to the Compare Data
- CONTAINS the Compare Data (Ignore Offset)

These five options allow branching based on the results of a previous *Tag Read Data* or *Tag Read ID* command.

#### **OR IF the previous READ/WRITE caused:**

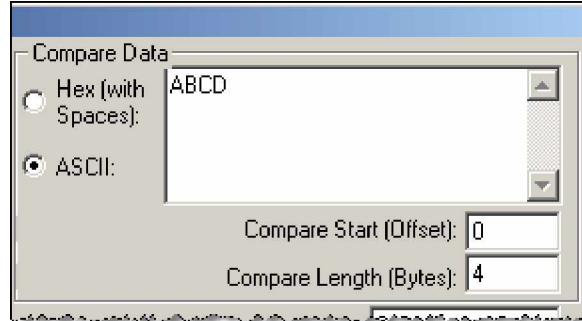
- Any Error
- The Following Error:
  - Partial Tag Fill Error
  - Partial Read Failure
  - Tag Not Found
  - Tag Not Found (Legacy)
  - General Syntax / Parameter Error



These two options allow branching based on errors generated by a previous RFID command (*Tag Search*, *Tag Read ID*, *Tag Read Data*, *Tag Write Data*, or *Tag Fill Data*).

### 2.1.18.2 Compare Data

On the right of the *Conditional Branch Options* dialog box, is a data entry box for entering a string of “*Compare Data*”, as well as fields for *Compare Start (Offset)* and *Compare Length (Bytes)*.



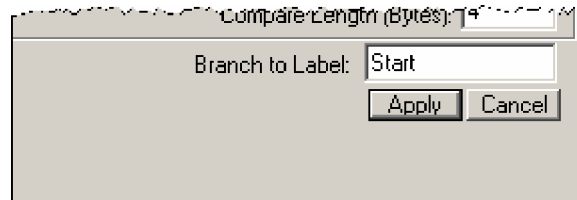
The Compare Start (Offset) is the offset within the last read result, not a tag address offset.

Therefore, if you just read 8 bytes from address 100 of the tag, and you want to compare the last two of those bytes, you would set the Compare Start (Offset) to six, and the Compare Length to two.

**NOTE:** All comparisons such as “*Less Than*” and “*Greater Than*” are compared byte for byte. Therefore, the string “*ABCD*” which has the actual HEX byte values of 41 42 43 44 is “*Less Than*” the string “*abcd*” which has the actual HEX byte values of 61 62 63 64. Likewise, the string “*Cat*” (43 61 74) is “*Greater Than*” the string “*CAT*” (43 41 54).

### 2.1.18.3 Branch to Label

Lastly, there is a “*Branch to Label*” field that identifies the label to which the macro will branch if the condition is met.



If the *Label* does not actually exist at the time you insert this command task, a warning message will be displayed next to the command item in the Macro Workspace that reads “(**\*\*\* NOT FOUND**).” This is an indication that C-Macro Builder does not yet know where you want to branch – you will still need to insert a label with the appropriate name.

**NOTE:** With all branch command tasks, care should be taken not to cause macros to enter “infinite loops,” unless that is the desired behavior.

**EXAMPLE 1:**

In example 1, eight bytes are read from a tag. If the first two bytes are “*Less Than*” the string “*XY*”, then the macro will use the *Conditional Branch* command task to branch to the Label “*Too\_Low*”, in which case Hub Output A is set, otherwise the macro ends.

```
Macro: "D:\EMS HF and Demo Applications\C-Macro_Builder\Example Macros\Too Low - Set Hub Output A.MSF" Length:27 [228 Left]
# Start
Tag Read (Length:8, Address:0, Timeout:2000ms)
IF last read (Length:2, Offset:0) < "XY" then BRANCH to "Too_Low"
END
# Too_Low
Set Hub Outputs {A }
END
Compiled Macro (Raw Hex): 00 07 05 00 00 00 08 07 D0 00 07 E2 00 14 02 00 58 59 00 00 00 03 EA 00 01 00 00
```

**EXAMPLE 2:**

In example 2, eight bytes are read from a tag. If the first 5 bytes are equal to the byte values *01 02 03 04 05*, the macro will use the *Conditional Branch* command task to branch to the Label “*MATCH*”, in which case Hub Output A is set, otherwise the macro ends.

```
Macro: "D:\EMS HF and Demo Applications\C-Macro_Builder\Example Macros\Match - Set Hub Output A.MSF" Length:30 [225 Left]
# Start
Tag Read (Length:8, Address:0, Timeout:2000ms)
IF last read (Length:5, Offset:0) = { 1 2 3 4 5 } then BRANCH to "Match"
END
# Match
Set Hub Outputs {A }
END
Compiled Macro (Raw Hex): 00 07 05 00 00 00 08 07 D0 00 0A E4 00 17 05 00 01 02 03 04 05 00 00 00 03 EA 00 01 00 00
```

**EXAMPLE 3:**

In example 3, 100 bytes are read from a tag. If the string “*FAILURE*” is found anywhere in the previous 100-byte read, the macro will use the *Conditional Branch* command task to branch to the Label “*FOUND\_FAILURE*” and will then set Hub Output A, otherwise the macro ends.

```
Macro: "D:\EMS HF and Demo Applications\C-Macro_Builder\Example Macros\Failure - Set Hub Output A.MSF" Length:32 [223 Left]
# Start
Tag Read (Length:100, Address:0, Timeout:2000ms)
IF last read CONTAINS "FAILURE" then BRANCH to "Found_Failure"
END
# Found_Failure
Set Hub Outputs {A }
END
Compiled Macro (Raw Hex): 00 07 05 00 00 00 64 07 D0 00 0C E4 00 19 07 FF 46 41 49 4C 55 52 45 00 00 00 03 EA 00 01 00 00
```

**EXAMPLE 4:**

In example 4, if the attempted *Tag Read* command task fails for any reason, the macro will execute the *Conditional Branch* command task and will branch to the Label "*Error\_Occurred*" thereby setting Output A on the Hub. If no error occurs, the macro will end normally after the Tag Read.

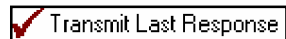
Notice that the "*On Error, Continue Macro*" command task is required in this example, otherwise the macro would end on an error and not set Hub Output A.

The screenshot displays the C-Macro Builder interface for a macro named "D:\NEMS HF and Demo Applications\C-Macro\_Builder\Example Macros\Error Occurred - Set Hub Output A.MSF". The macro steps are as follows:

- Start** (Green circle icon):
  - On Error, Continue Macro (Wrench icon)
  - Tag Read (Length:20, Address:100, Timeout:2000ms) (Blue arrow icon)
  - IF last RF command caused an error then BRANCH to "Error\_Occurred" (Blue diamond icon)
  - END (Red circle icon)
- Error\_Occurred** (Green circle icon):
  - Set Hub Outputs {A B } (Green lightning bolt icon)
  - END (Red circle icon)

At the bottom, the compiled macro is shown in raw hex: 00 02 E6 00 00 07 05 00 64 00 14 07 D0 00 03 E7 00 14 00 00 00 03 EA 00 03 00 00

## 2.1.19 Transmit Last Response



Transmit Last Response

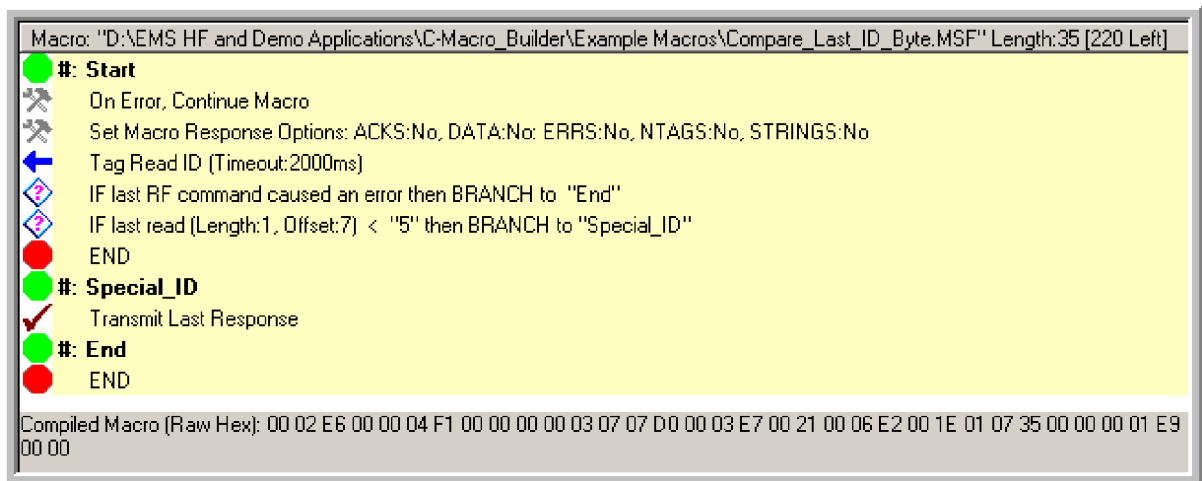
This command task causes the last result/response of a *Tag Search*, *Tag Read ID*, *Tag Read Data*, *Tag Write Data* or *Tag Fill Data* operation to be sent (or re-sent) to the host.

This command task is typically used when a previously inserted *Macro Response Options* command task has disabled one or more responses, thereby allowing the macro to compare the results before determining if the result should actually be sent back to the host. This allows filtering of responses, and can make applications function more efficiently.

When the item is inserted into a macro, no dialog box appears because there are no editable items for this command task.

### EXAMPLE:

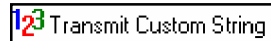
In this example, an 8-byte tag ID is read, and if the last byte is less than “5”, then (and only then) is the tag ID sent to the host, via the “*Transmit Last Response*” command task.



**NOTE:** When the *Transmit Last Response* command task is executed, the current “Macro Response Options” are ignored. This command task will transmit the last response even when all macro responses are disabled.



## 2.1.20 Transmit Custom String

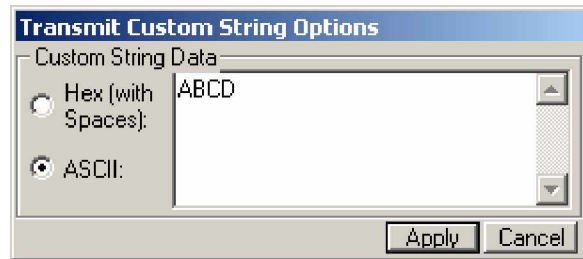


This command task instructs the controller to transmit a custom string of data back to the host.

The *Transmit Custom String* command task allows additional flexibility in what is sent back to the host, and provides the ability to send informative data in addition to, or instead of tag data.

This command task is also helpful in debugging macros, allowing users to trace the execution of the macro, by inserting several “*Transmit Custom String*” command tasks at various points in the macro.

When the item is inserted into a macro, the *Transmit Custom String Options* dialog box is displayed.

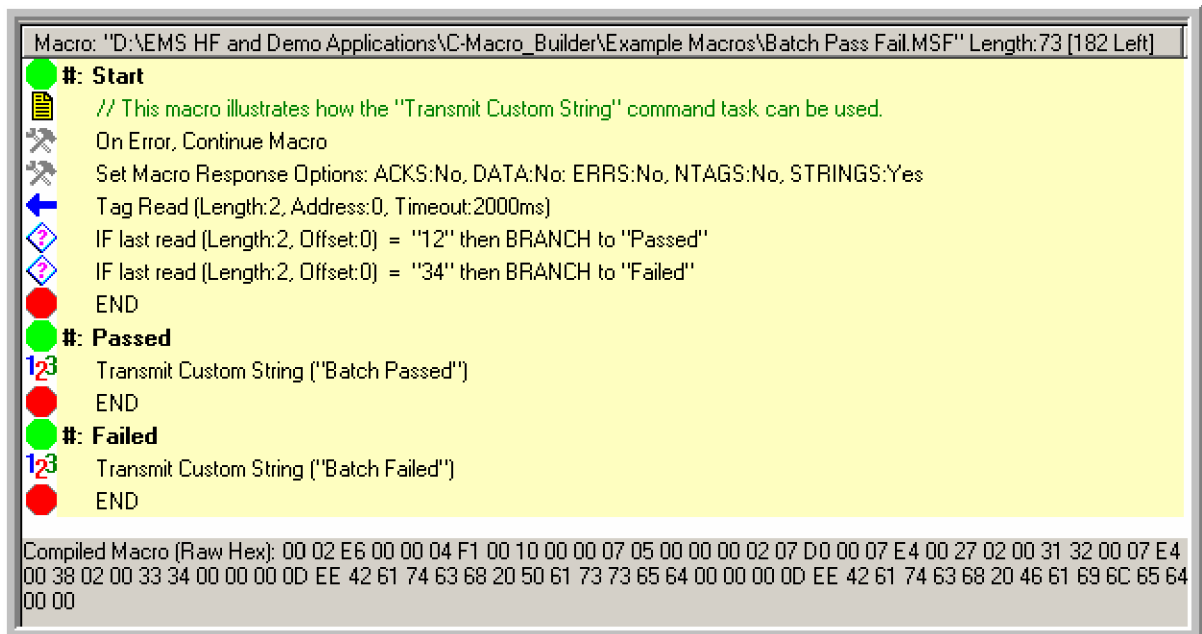


The editable options for this command task are:

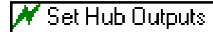
**Custom String Data:** (0 to 247 bytes), data can be displayed in either *HEX* (with Spaces) or *ASCII*.

### EXAMPLE 1:

In the following example, two bytes are read from a tag. If those two bytes are “01 02”, then the custom string “*Batch Passed*” is sent to the host. If the two bytes are “03 04” then the custom sting “*Batch Failed*” is sent to the host.



### 2.1.21 Set Hub Outputs

 **Set Hub Outputs** This command task causes the controller to send a packet to a Subnet16 Hub, instructing it to set (enable) one or more of the four Hub Outputs. This allows a macro to directly control external hardware, such as conveyors, diverters, stack lights, etc.

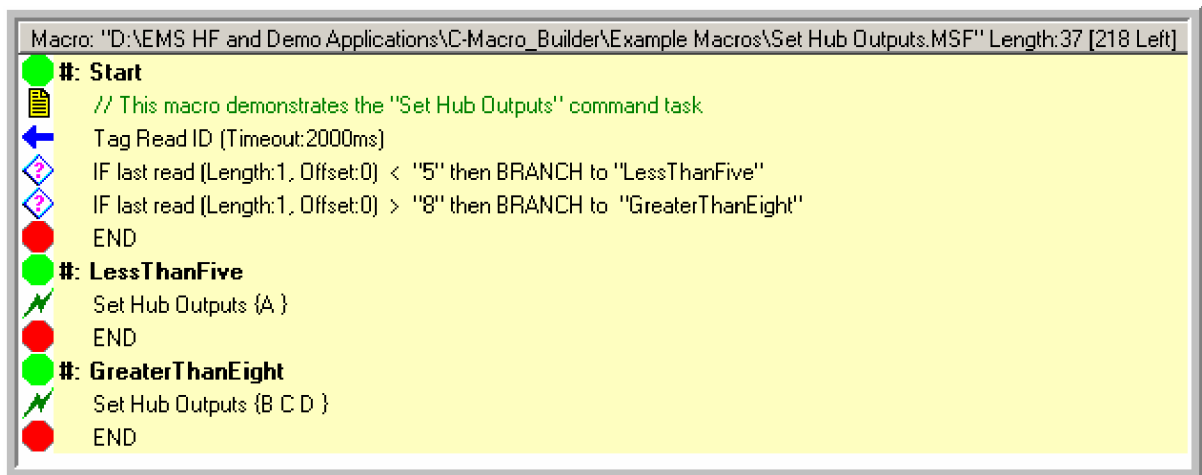
When the item is inserted into a macro, the *Set Hub Outputs Options* dialog box is displayed.




Any or all of the four Hub Outputs may be set. If a Set Hub Output checkbox is not checked, it will NOT be set when this command task is executed (therefore Hub Outputs already set by some other means will remain set). Only those Hub Outputs explicitly set using this command task will be affected.

**EXAMPLE:**

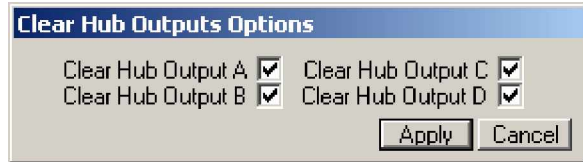
In the following example, a tag ID is read. If the first digit of the ID is less than 05, then Hub Output A is set. If the first digit is greater than 08, then Hub Outputs B, C and D are set.



## 2.1.22 Clear Hub Outputs

 **Clear Hub Outputs** This command task causes the controller to send a packet to a Subnet16 Hub, instructing it to clear (disable) one or more of the four Hub Outputs. This allows a macro to directly control external hardware, such as conveyors, diverters, stack lights, etc.

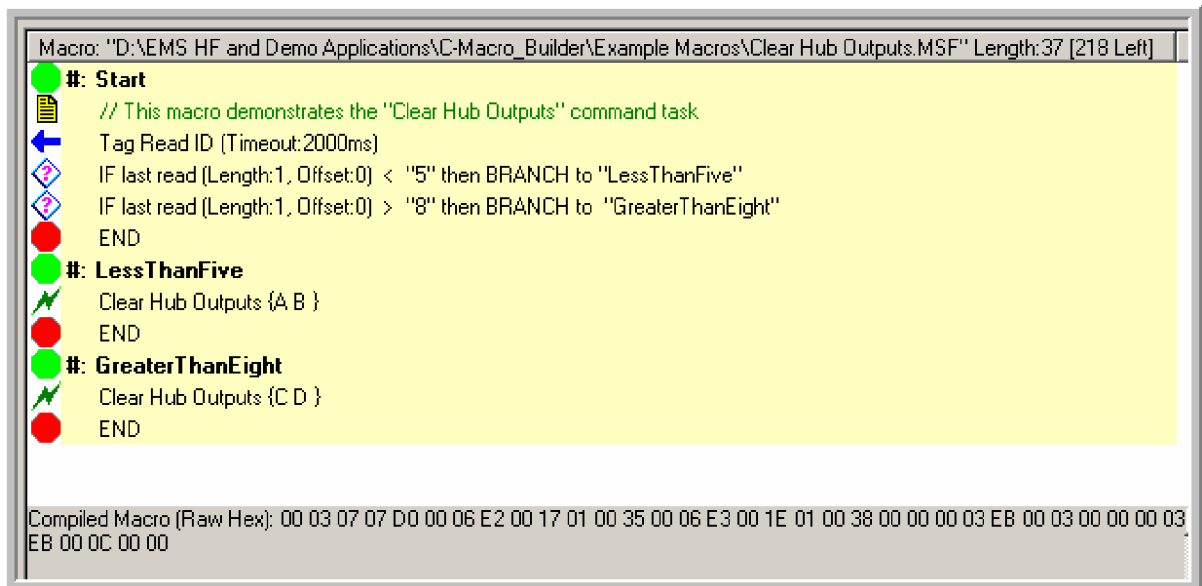
When the item is inserted into a macro, the *Clear Hub Outputs Options* dialog box is displayed.



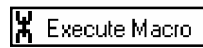
Any or all of the four Hub Outputs may be cleared. If a Clear Hub Output checkbox is not checked, it will NOT be cleared when this command task is executed. Only those Hub Outputs explicitly cleared using this command task will be affected.

### EXAMPLE:

In the following example, a tag ID is read. If the first digit of the ID is less than 05, then Hub Output A and B are cleared. If the first digit is greater than 08, then Hub Outputs C and D are cleared.

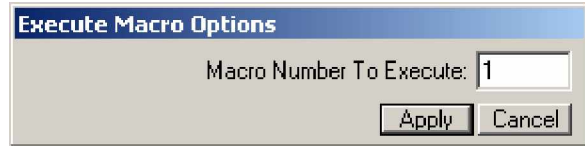


### 2.1.23 Execute Macro



This command task causes the controller to end the current macro and immediately begin executing another macro. When that macro finishes, control is **not** returned to the previous macro. This command task allows macros to be chained together to form larger programs.

When the item is inserted into a macro, the *Execute Macro Options* dialog box is displayed.



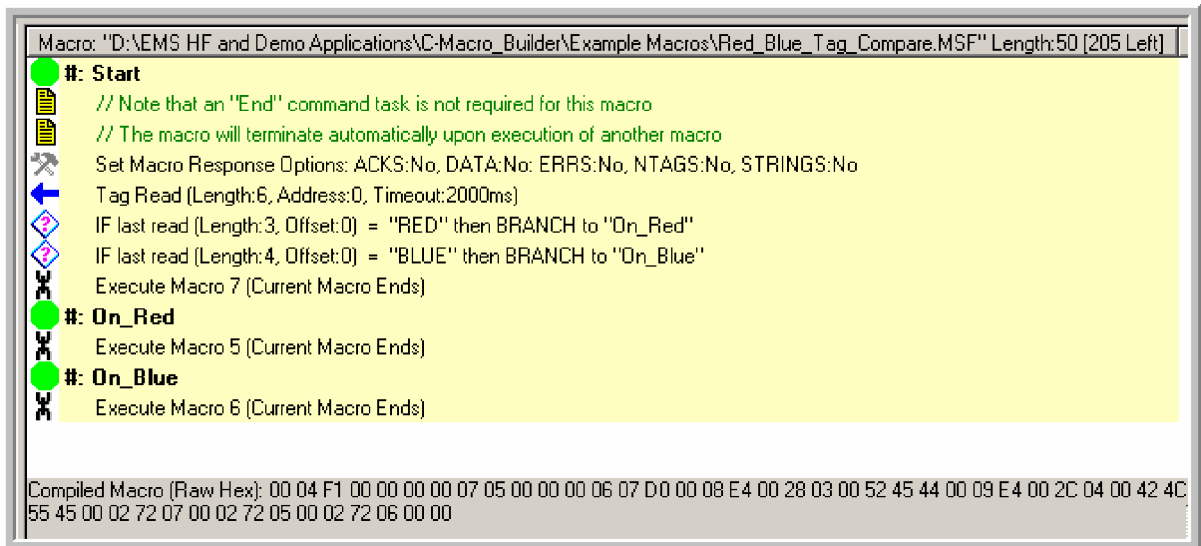
Any macro number (between 1 and 8) can be executed with this command task.

Compiled macros do not have an inherent 'macro number' associated with them – they can be loaded into any of the eight macro slots available in supported EMS RFID controllers. Therefore, it is up to the user to keep track of which macros are loaded to which macro slots.

**NOTE:** It is possible for a macro to execute itself, and care should be taken not to cause macros to enter "infinite loops" in this manner unless that is the desired behavior. If a macro is instructed to execute another macro that does not exist, the current macro execution will terminate.

**EXAMPLE:**

In the following example, the first 6 bytes of a tag are read. If the first 3 bytes = "RED" then Macro 5 is executed. If the first 4 bytes = "BLUE" then Macro 6 is executed. Otherwise, Macro 7 is executed.

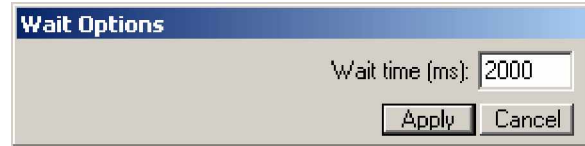


## 2.1.24 Wait



This command task causes the controller to pause during the execution of a macro. When the item is inserted into a macro, the *Wait Options* dialog box is displayed.

The controller will wait for the length of time specified in the *Wait Time* field, before continuing to execute the remainder of the macro.



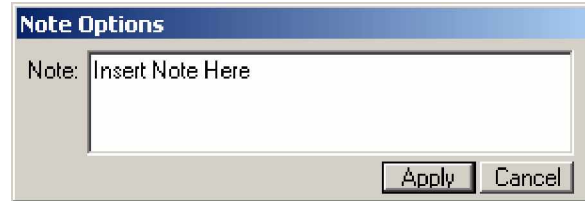
### 2.1.25 Note



This “command task” is actually a location to insert text notations within a macro, which is used for comments or other informational purposes.

Notes do not affect the flow of the macro execution and can include descriptions of program logic, version numbers, or anything that the user feels should be noted.

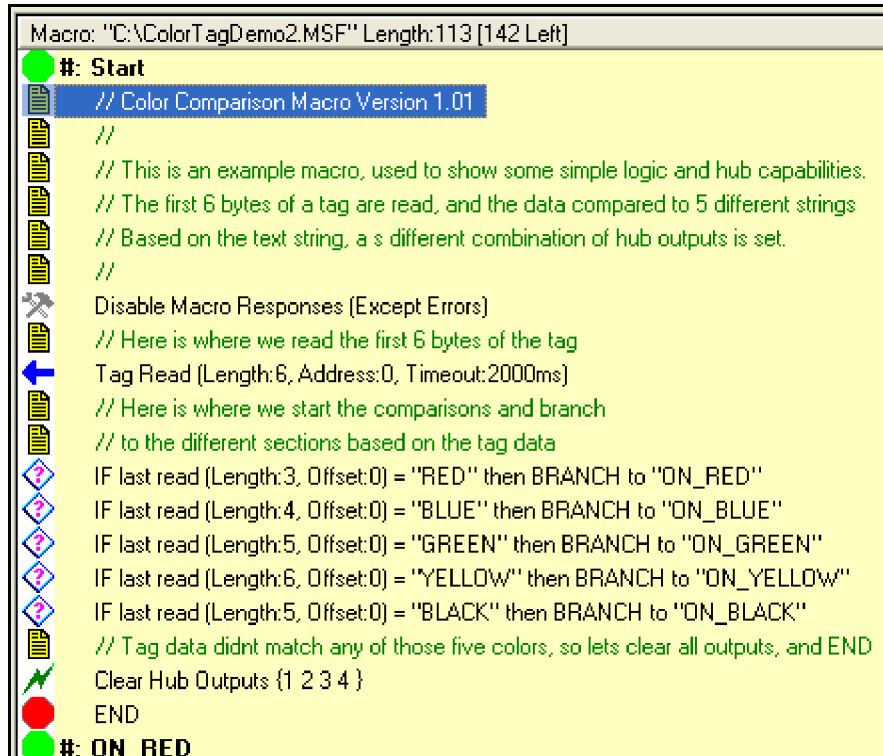
When the item is inserted into a macro, the *Note Options* dialog box is displayed.



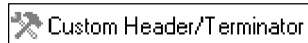
Notes can contain up to 255 characters each and do not take up any macro space.

#### EXAMPLE:

The following example displays a portion of a large macro that contains several “Notes” that are used to describe the flow of the macro, indicate the version number and provide other information.



## 2.1.26 Custom Header/Terminator



This command task, for serial-based controllers only, is used in conjunction with the *Custom* option in the *Macro Response Options* command task. When one of the five types of host-bound responses is set to *Custom* in the *Macro Response Options* command task dialog, this command task can then be used to define a custom header and terminator for the specified response type.

The five response types are:

- Command ACK Responses
- Tag Data Responses
- Error Responses
- Multi-Tag NTAGS Responses
- Custom String Responses

When this command task item is inserted into a macro, the *Custom Header / Terminator Options* dialog box is displayed.

Select one of the five response type options and then enter your custom header and terminator data in the provided fields. The two length parameters will be automatically populated based on the number of bytes entered into their corresponding fields.

(See [Section 2.1.2](#) for descriptions of the five response types).

**Custom Header/Terminator Options\***

Set Custom Header/Terminator for:

Command ACK Responses:

Tag Data Responses:

Error Responses:

Multitag NTAGS Responses:

Custom String Responses:

Custom Header/Terminator

Header: AA

Header Length (Bytes): 2

Hex (with Spaces):

ASCII:

Terminator: ZZ

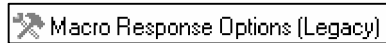
Terminator Length (Bytes): 2

\* Serial Communication Only

Apply Cancel

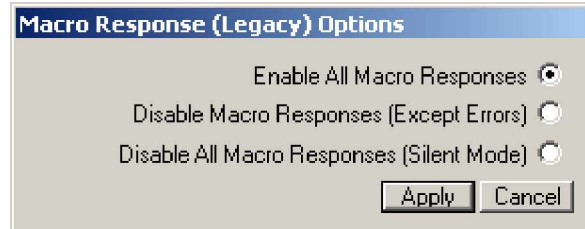
**NOTE:** When using a custom header and/or terminator, the normal ABx Fast header and terminator bytes will be replaced by user-defined data. Therefore, users will no longer be able to communicate with the controller via the *Cobalt HF Serial Dashboard* software utility.

## 2.1.27 Macro Response Options (Legacy)



This command task allows the user to specify when and if they want macro responses delivered to the host.

This command task is a legacy “hold-over” feature from previous versions of C-Macro Builder. It is recommended that users utilize the *Macro Response Options* command task instead, as it includes the same features as the *Response Options* command task, yet provides enhanced functionality.



When the item is inserted into a macro, the *Macro Response Options* dialog box is displayed.

- Enable All Macro Responses
- Disable Macro Response (Except Errors)
- Disable All Macro Responses (Silent Mode)

**NOTE:** This command task is used by HF-0405-Series RFID controller instead of the “*Macro Response Options*” command task, which is not supported by HF-0405 controllers. All other Cobalt controllers can use either response option command.



## 2.1.28 End



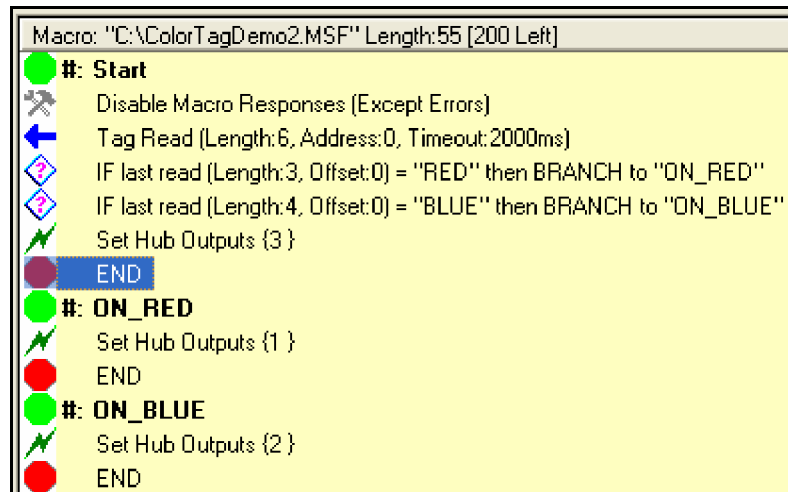
The **End** command task causes a macro to immediately terminate. This command task can be inserted in a macro at any point where macro execution is to stop.

There are no editable items for this command task, therefore, when the item is inserted into a macro, no dialog box is displayed.

C-Macro Builder automatically inserts an implied “End” command task at the conclusion of every compiled macro, regardless of whether or not one has been manually inserted in the macro source code. Still, it is recommended that users place an “End” command task at the end of all macros.

### EXAMPLE 1:

In the following example, the first 6 bytes of a tag are read. If the first 3 bytes = “RED,” then Hub output 1 is set, and the macro ends. If the first 4 bytes = “BLUE,” then Hub output 2 is set, and the macro ends. Otherwise, Hub output 3 is set, and the macro ends.

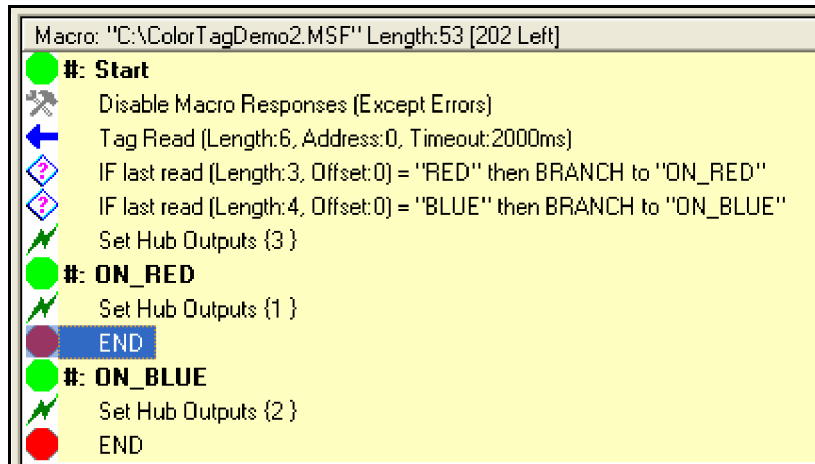


**EXAMPLE 2:**

The following example illustrates a common programming error.

In this example, six bytes from a tag are read. If the first bytes of the tag do not match "RED" or "BLUE", Hub output 3 is set. However, macro execution does not stop after Hub output 3 is set, and the macro will continue executing, passing over the "ON\_RED" label, and setting Hub output 1, before ending.

An "End" command task should have been inserted after the *Set Hub Outputs {3}* command task to stop the macro at that point.



170 TECHNOLOGY CIRCLE  
SCOTTS VALLEY, CA 95066 USA  
TELEPHONE: (831) 438-7000  
FAX: (831) 438-5768  
WEBSITE: WWW.EMS-RFID.COM  
EMAIL: INFO@EMS-RFID.COM