Handling multiple motor/PFC parameters with iMOTION™

A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution Designer

About this document

Scope and purpose

iMOTION™ is a turnkey yet versatile solution for efficient control of variable speed motors and PFC optionally. We explain how to set up one iMOTION™ to handle multiple parameter sets for motors and PFC optionally.

Intended audience

This application note is targeting engineers developing variable speed motor control drives using iMOTION™.

Table of contents

About this document ......................................................................................................................................................... 1
Table of contents ............................................................................................................................................................. 1
1 Introduction ................................................................................................................................................................. 2
2 Multiple Parameter Handling by MCEWizard/Designer ......................................................................................... 3
   2.1 Multiple-parameter block definition .................................................................................................................. 3
   2.1.1 MCEWizard configuration .......................................................................................................................... 3
   2.1.2 Motor ID using UART .................................................................................................................................. 6
   2.1.3 Motor ID using GPIO pins ........................................................................................................................... 8
   2.1.4 MotorID using analog input ........................................................................................................................ 9
   2.1.5 Motor ID using Direct Select ....................................................................................................................... 9
   2.2 Multiple-parameter application example .......................................................................................................... 10
      2.2.1 Preparation of MCEWizard .txt parameter files ...................................................................................... 10
      2.2.2 Combine .txt parameter files into a single .ldf file .................................................................................. 12
      2.2.3 Final combine .ldf file to be used with MCEDesigner .............................................................................. 19
      2.3 Programing combined files with the MCEDesigner ..................................................................................... 19
3 Multiple Parameter Handling by iMOTION™ Solution Designer .......................................................................... 20
   3.1 Multiple Parameter Set Definition .................................................................................................................... 20
      3.1.1 iSD Configuration ....................................................................................................................................... 20
      3.1.2 Motor ID using UART .................................................................................................................................. 22
      3.1.3 Motor ID using GPIO pins ........................................................................................................................ 22
      3.1.4 MotorID using analog input ....................................................................................................................... 22
      3.1.5 Build Project with Multiple Parameter Sets ............................................................................................ 22
4 References ............................................................................................................................................................... 26
5 Revision history .......................................................................................................................................................... 27
1 Introduction

iMOTION™ offers users greater control over permanent magnet motors by integrating an advanced software implementation in various hardware form factors. iMOTION™ comes with two development tools: the MCEWizard and the MCEDesigner. The MCEWizard is a GUI tool that allows users to enter motor or PFC parameters to control a motor and PFC optionally such as motor stator inductance and resistance, back electromotive Force, and motor speed limits.

For certain applications, users may need to utilize the same iMOTION™ control solution to run either different motors or the same motor with PFC, optionally under different load conditions. As an example, one iMOTION™ solution should be able to run a compressor under the right cooling profile in any of three refrigerator models of different volumetric capacities. Separately, a fan manufacturer may use the same iMOTION™ solution to control three motor models of varying power ratings and fan blade sizes.

In this application note, we will describe how to set up multiple parameter sets for motors and PFC optionally in order to support controlling various motors and hardware configurations with one solution. We will also describe how to set up multiple parameter sets by using new development tool iMOTION™ Solution Designer (iSD).
2 Multiple Parameter Handling by MCEWizard/Designer

2.1 Multiple-parameter block definition

2.1.1 MCEWizard configuration

In this application note, we look at an example of running four different motor/PFC and hardware configurations with a single iMOTION™ controller. We define each motor/PFC & hardware configuration with a set of parameters. In this example, we will have four parameter sets, from which our iMOTION™ controller can select.

By default, only one set of parameters is stored in FLASH and loaded into RAM of an iMOTION™ device, and the ability to select different parameter sets is disabled in the MCEWizard. We can enable multiple-parameter set support via the Multiple Motor Parameter Set Support option in the MCEWizard, as shown in Figure 1.

Users can use one of the following ways to tell MCE the desired parameter set number upon start-up:

- User UART
- PAR0 / PAR1 / PAR2 / PAR3 (GPIO) pins
- PARAM (AIN) pin
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

- Direct select

Each parameter set can be assigned with a unique parameter set number that can be specified in the MCEWizard. When the feature is enabled, the iMOTION™ firmware will load the parameter set with the matching parameter set number. To set up this ID, choose a number to enter in the “Parameter Set Number” section for the set of parameters entered in the MCEWizard, as shown in Figure 2. Detail setting will be described in section 2.2.1.

![Parameter Set Number in MCEWizard](image)

Figure 2 Parameter Set Number in MCEWizard

When working with different setups and iMOTION™ devices, the MCEWizard will use 4 kilobytes of flash memory to store various control parameter data. There are 16 parameter blocks, each of them being 256 bytes in size. A maximum of 15 parameter sets can be programmed in order to support different motor types or hardware, and one block is reserved to store system parameters.

For a system with only a motor control function, each parameter set will take one parameter block. In this case, the valid parameter set IDs can range from 0 to 14.

For a system with motor control and PFC functions, each parameter set will take two consecutive parameter blocks. The motor control parameter set will be stored in the selected parameter block, and the PFC parameter set will be stored in the subsequent parameter block. In this case, the valid parameter set IDs are 0, 2, 4, 6, 8, 10, and 12, with only 7 different configurations made available.

The MCEWizard output (*.txt) that contains the parameter values and the specified parameter set number will be generated. Section 3 describes how to combine all these parameter files into one .ldf file for easy loading into the iMOTION™ device with the MCEDesigner.

Note: The available parameter selection methods depend on the iMOTION™ device used. IMM101T-046, for example, only supports Direct and User UART selection methods. IMC101T-T038 supports all four methods.

Revision 1.1
2023-06-01
In any case, when one of the four selection methods are specified, the register ParPageConf is updated automatically by the MCEWizard, and enables one of the four solutions. Figure 3 shows the full parameter loading procedure for iMOTION™ devices.

Figure 3  iMOTION™ parameter loading procedure

ParPageConf is a 16-bit register that defines the parameter-block selection method and default parameter block to upload. It is defined as follows:

[3:0] Parameter page selection:

0 - No selection
1 - Parameter page selection via UART
2 - Parameter page selection via analog Input
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

3- Parameter page selection via digital Input

[7:4] Default parameter page number

[15:8] Reserved

Section 2.2 describes in detail how to use this very important register in order to combine all .txt files into one .txt file to be loaded into the MCEDesigner.

2.1.2 Motor ID using UART

Specific Universal Asynchronous Receiver/Transmitter (UART) messages are defined to load the parameter block from flash to RAM, and save the parameter set from RAM to flash. 'Load parameter' command = 0x20 loads all parameters of one block into the dedicated RAM locations. If this method is chosen, ParPageConf [3:0] = 1.

The 'Load parameter' command loads the parameters from the specified parameter set stored in FLASH into the RAM. The valid range of the parameter set number is 0-14 when PFC function is disabled and 0, 2, 4, 6, 8, 10, or 12 during PFC function is enabled. If an odd number (e.g. 1) is selected when the PFC function is enabled, parameter load fault occurs and the specified parameter set is not loaded into RAM.

The 'Save parameter' command erases the selected parameter set in FLASH first and saves the parameters of the specified App ID to this parameter set in FLASH. The valid range of the parameter set number is 0-14 when PFC function is disabled, and 0, 2, 4, 6, 8, 10, or 12 when PFC function is enabled. If an odd number (e.g. 1) is selected when the PFC function is enabled, parameter load fault occurs and the specified parameter set is not loaded to RAM. The valid App ID value is 1 or 3. In the reply frame, data 0 word contains the value of 'Status' (0: success, 1: fail, 2: parameter set number not supported).

Figure 4  UART Load parameter command = 0x20, Data Word0 = 0x0020

Figure 5  UART Save parameter command = 0x21, Data Word0 = 0x0021
Here is an analysis of this data frame for loading or saving parameters:

- **Node address** is the first byte in a data frame. It is designed to allow one master to control multiple slaves in the same network. Each slave node has its unique node ID. The slave only acknowledges and responds to the message with the same ID. Two broadcast addresses (0x00 and 0xFF) are defined for different usages. If a message is received with address=0x00, all the slaves execute the command, but will not send a reply to the master. This is useful in a multiple-slave network, and the master needs to control all the slaves at the same time, like turn on all the motors by sending only one message. If a frame with address=0xFF is received, the slave will execute the command and send a reply to the master. This is useful in a 1-to-1 configuration when the master does not know or does not need to know the slave node address.

- **Command** is the specific UART command to load or save a created set of parameters. Command = 0x20

- **Data Word 0** describes if the process is loading parameter 0x0020 or saving parameter 0x0021

- **Dataword 1** LSB = 0x00 and MSB = chosen parameter set number corresponding to the correct one from among 15.

- **Checksum** is 16-bit format that shall be calculated as below: [Command: Node address] + Data Word 0 + Data Word 1 + Checksum = 0x0000

Here is an example of where to use UART command to load parameter block 3:

Input : Node address = 1, Command = 0x20, Data Word 0 = 0021, Data Word 1 = 0x0003

[Command:Node address] = 0x2001

Checksum = -1 x ( 0x2001 + 0021 + 0003 ) = 5202

UART message to send to iMOTION™ controller to load parameter block 3: 01 20 0021 0003 5202
2.1.3 Motor ID using GPIO pins

This parameter block is selected based on the input of the four General Purpose Input/Output (GPIO) pins. The GPIO pins used for parameter set selection are named as “PAR0,” “PAR1,” “PAR2,” and “PAR3.” Mapping between parameter page selections based on GPIO pins are listed in Table 1. If this method is chosen, ParPageConf [3:0] = 3.

Any number from 0 to 14 can be selected when PFC function is disabled. Only even numbers equal or smaller than 12 (0, 2, 4, 6, 8, 10, or 12) can be selected when PFC function is enabled. If any other number is selected when PFC is enabled, parameter load fault occurs and the specified parameter set is not loaded into RAM.

<table>
<thead>
<tr>
<th>GPIO Input</th>
<th>Parameter block</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>2</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>3</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>4</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>5</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>6</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>7</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>8</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>9</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>10</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>11</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>12</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>13</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 1 Parameter page selection with GPIOs

When the GPIO selection method is not being used, these GPIOs are available to be used in script code as shown in Figure 7.
### 2.1.4 MotorID using analog input

If this method is chosen, ParPageConf [3:0] = 2. Parameter block is selected based on the analog input voltage level of the ‘PARAM’ pin of an iMOTION™ device. Mapping between parameter page and analog input voltage is shown below:

\[
ParameterBlock = \text{Integer}\left(\frac{\text{AnalogInput}}{V_{\text{adcref}}} \times 15\right)
\]

For example, if analog input at PARAM pin is 1.2 V and \(V_{\text{adcref}}=3.3\) V, parameter block number =5.

Any number from 0 to 14 can be selected when PFC function is disabled. Only even numbers equal or smaller than 12 (0, 2, 4, 6, 8, 10, or 12) can be selected when PFC function is enabled. If any other number is selected when PFC is enabled, parameter load fault occurs and the specified parameter set is not loaded into RAM.

### 2.1.5 Motor ID using Direct Select

The parameters’ block selection is based on the “ParPageConf [7:4]” parameter bit field value. This requires a manual bit field update in the MCEDesigner or parameter file. If we want to load parameter block 3, set ParPageConf as follows:

2.2 Multiple-parameter application example

In the previous section, we demonstrated how to set up the MCEWizard and how to use the four different ways of selecting a parameter set. In this chapter, we describe a method of combining and loading four different parameter sets into the controller using the MCEWizard and the MCEDesigner.

2.2.1 Preparation of MCEWizard .txt parameter files

As there are four different configurations to select from, via UART with a single iMOTION™ device, we first need to generate one parameter file for each configuration. We use the MCEWizard to create all the parameter files. Each file and parameter set are defined by a unique Parameter Set Number.

2.2.1.1 PFC function: disabled

In this example, we enter 0, 1, 2, and 3 as the Parameter Set Number for each configuration.

![Image of MCEWizard interface showing parameter block 3](image)

**Figure 8** Example for parameter block 3 in MCEWizard.

This should create four different .txt files once compiled with the main identification registers as follows:

<table>
<thead>
<tr>
<th>Parameter File Name (Example)</th>
<th>Parameter Set Number</th>
<th>Set Support</th>
<th>##MOTOR1_REGS</th>
<th>ParPageConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterFile0.txt</td>
<td>0</td>
<td></td>
<td>0</td>
<td>2049</td>
</tr>
<tr>
<td>ParameterFile1.txt</td>
<td>1</td>
<td>Motor-ID from UART</td>
<td>1</td>
<td>2065</td>
</tr>
<tr>
<td>ParameterFile2.txt</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2081</td>
</tr>
<tr>
<td>ParameterFile3.txt</td>
<td>3</td>
<td></td>
<td>3</td>
<td>2097</td>
</tr>
</tbody>
</table>
2.2.1.2 PFC function: enabled

In this example, we enter 0, 2, 4, and 6 as the Parameter Set Number for each configuration.

![Image showing MCEWizard settings for Parameter Set Number]

Figure 9 Example for parameter block 3 in MCEWizard.

This should create four different .txt files once compiled with the main identification registers as follows:

<table>
<thead>
<tr>
<th>Parameter File Name (Example)</th>
<th>Parameter Set Number</th>
<th>Set Support</th>
<th>##MOTOR1_REGS</th>
<th>##PFC_REGS</th>
<th>ParPageConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterFile0.txt</td>
<td>0</td>
<td>Motor-ID from UART</td>
<td>0</td>
<td>1</td>
<td>2049</td>
</tr>
<tr>
<td>ParameterFile1.txt</td>
<td>2</td>
<td></td>
<td>2</td>
<td>3</td>
<td>2081</td>
</tr>
<tr>
<td>ParameterFile2.txt</td>
<td>4</td>
<td></td>
<td>4</td>
<td>5</td>
<td>2113</td>
</tr>
<tr>
<td>ParameterFile3.txt</td>
<td>6</td>
<td></td>
<td>6</td>
<td>7</td>
<td>2145</td>
</tr>
</tbody>
</table>

Now that there are four different .txt files with proper identification, users can combine them into a single file that can be uploaded to the MCEDesigner.
2.2.2 Combine .txt parameter files into a single .ldf file

The first thing to do is to use the MCEWizard “Combine Files Page” functionality:

1. For the Parameters File, choose the first one generated. In our example, it will be ParameterFile0.txt
2. For the MCE Firmware File, choose the corresponding .ldf file. In our example, it will be IMC101T-046
3. Create a Combine File. In this example, we will call it Parameter0.ldf

Repeat steps 1 to 3 for the number of parameters you desire. In our example, we will do this exercise four times.

Figure 10 Combine Files Page in MCEWizard.

Figure 11 shows an example of a combined .ldf file that is generated by the operation above. This file consists of three sections when PFC is disabled, and four sections when PFC is enabled.
Handler multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

Figure 11  Example of combined .ldf file

Use the following steps to combine multiple .ldf files:

1. Extract motor parameter section (and PFC parameter section, if necessary)
2. Change Page number
   - The page number is defined in the line starting from "a0 22" after "% Erase Parameter Set" and "a0 21" after "% Check Parameter Set". The third value in this line is the page number.
   - If PFC is disabled, the page number will start from 0 and is incremented. If PFC is enabled, the page number will be an even number starting from 0 for motor parameter section, and an odd number starting from 1 for PFC parameter section.
3. Repeat (1) and (2) for all combined .ldf files.
4. Combine all modified .ldf files.
5. Add Firmware section in the beginning of combined .ldf file made in (4), and add system parameter section at the end of the combined .ldf file made in (4).
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

VBA function to combine multiple .ldf combined files into a single usable one in MCEWizard

The procedure above would be time consuming if done manually. That is why a simple Excel VBA function is provided, which can extract all the useful information from the .ldf files and combine it into one final .ldf file. The final file will have all the different parameter block sets, and can easily be uploaded with the MCEDesigner.

VBA code is shown in Code Listing 1 to Code Listing 3. Please copy and paste all codes and create a macro "Make_Combined_LDF" in MS Excel to run it.

Code Listing 1  Main Function Make_Combined_LDF()

```vba
001: Sub Make_Combined_LDF()
002:     Dim ChDir As String, f As Variant
003:     Dim i As Integer, n As Integer
004:     Dim rc As Integer
005:     Dim if_success As Boolean
006:     Fin = Application.GetOpenFilename(FileFilter:="LDF file(*.ldf),*.ldf", MultiSelect:=True)
007:     If VarType(Fin) = vbBoolean Then
008:         MsgBox ("Cancelled")
009:     Else
010:         If IsArray(Fin) Then
011:             n = UBound(Fin)
012:             If n > 15 Then
013:                 MsgBox "# of files is more than 15.", vbExclamation
014:             Else
015:                 Fout = Application.GetSaveAsFilename(FileFilter:="LDF file(*.ldf),*.ldf", Title:="Save combined LDF file")
016:                 If Dir(Fout) <> "" Then
017:                     If Dir(Fout) <> "" Then
018:                         rc = MsgBox("File exist. Overwrite?", vbYesNo + vbQuestion)
019:                         If rc = vbNo Then
020:                             MsgBox ("Quit the process")
021:                         End If
022:                     End If
023:                 End If
024:     End If
025:     ' Check PFC
026:     page_count = count_pages(Fin(1))
027:     If (page_count = 3) Then
028:         MsgBox ("PFC function enabled")
029:     Else
030:         MsgBox ("PFC function disabled")
031:     End If
032:     Open Fout For Output As #2
033:     For i = 1 To n
034:         if_success = Make_LDF(Fin(i), i)
035:         If if_success = False Then
036:             MsgBox ("Combined fine generation FAILED.")
037:             Exit For
038:         End If
039: Next i
040: Close #2
```
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

Code Listing 1  Main Function Make_Combined_LDF()

```vbnet
040: If if_success = False Then
041: MsgBox ("Combined fine generation FAILED.")
042: Else
043: MsgBox ("Combine complete!")
044: End If
045: End If
046: Else
047: MsgBox ("Selected file is only one and no process will be executed")
048: End If
049: End If
050: End Sub
051:
052: Function Make_LDF(f As Variant, i As Integer) As Boolean
```

Code Listing 2  Function Make_LDF()

```vbnet
001: Function Make_LDF(f As Variant, i As Integer) As Boolean
002: Dim Ftemp As String, s As String
003: Dim file_operation_success As Boolean
004: file_operation_success = True
005: If page_count <> count_pages(f) Then
006: MsgBox ("Both PFC enabled and disabled are included in the file list.")
007: file_operation_success = False
008: Else
009: Open f For Input As #1
010: Do Until EOF(1)
011: Line Input #1, s
012: If InStr(s, "Parameters Data Section Begin") Then
013: Do
014: Line Input #1, s
015: Print #2, s
016: If InStr(s, "% Check Parameter Set") Then
017: Line Input #1, s
018: Print #2, s
019: Line Input #1, s
020: Print #2, s
021: If page_count = 3 Then
022: Do
023: Line Input #1, s
024: Print #2, s
025: If InStr(s, "% Check Parameter Set") Then
026: Line Input #1, s
027: Print #2, s
028: Exit Do
029: End If
030: Loop
031: End If
032: If i = UBound(Fin) Then
```
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution
Multiple Parameter Handling by MCEWizard/Designer

Code Listing 2  Function Make_LDF()

```plaintext
034:  Do
035:     Line Input #1, s
036:     Print #2, s
037:     If InStr(s, "% Check Parameter Set") Then
038:         Line Input #1, s
039:         Print #2, s
040:         Exit Do
041:     End If
042:     Loop
043:     End If
044:     Exit Do
045:     End If
046:     Loop
047:     End If
048:     Loop
049:     Close #1
050:     End If
051:     Make_LDF = file_operation_success
052:     End Function
```

Code Listing 3  Function count_pages()

```plaintext
001:  Function count_pages(f As Variant) As Integer
002:    Dim count As Integer
003:    Dim s As String
004:
005:    count = 0
006:    Open f For Input As #1
007:    Do Until EOF(1)
008:        Line Input #1, s
009:        If InStr(s, "% Page") And InStr(s, "AppID") Then
010:            count = count + 1
011:        End If
012:    Loop
013:    Close #1
014:    count_pages = count
015:    End Function
```

A macro containing this function can now be assigned to a button (Make_Combined_LDF, for example, to merge all the .ldf files into a single one that can then be uploaded to MCEDesigner.

When PFC function is disabled, the macro extracts the content of Page 00 - AppID 01 (Motor Control parameters) and Page 0f - AppID 00 (System Control parameters) from each .ldf file, and combines them into a new .ldf file containing all four parameter sets as well as the system control parameters. Figure 12 shows this newly combined .ldf file containing all the parameter sets:
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

The new .ldf file should have an AppID 01 section containing different parameter files and 1 AppID containing the system parameter.

When PFC function is enabled, the macro extracts the Page 00 - AppID 01 (Motor Control parameters), Page 01 - AppID 03 (PFC control parameters), and Page 0f - AppID 00 (System Control parameters) from each .ldf file, and combines them into a new .ldf file containing all four parameter sets as well as the system parameters. Figure 13 shows this new combined .ldf file containing all the parameter sets:
Handling multiple motor/PFC parameters with iMOTION™
A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

Multiple Parameter Handling by MCEWizard/Designer

Figure 13  Combined Parameter.ldf file detail for four different sets of parameters (partial).

The new .ldf file should have an AppID 01 section containing different parameter sets for motor control, an AppID 03 section containing different parameter sets for PFC function, and 1 AppID 0f containing the system parameters.

In our example, there is now a single .ldf file containing four different parameter sets.
2.2.3 Final combine .ldf file to be used with MCEDesigner

The last step is to reproduce the exercise described in section 3.2 to create a final combined file containing all different parameter sets and the desired firmware into a single .ldf file.

**Figure 14** Last step into creating a combined .ldf file containing all different parameter sets in MCEWizard.

2.3 Programing combined files with the MCEDesigner

Now that we have our combined file containing all our needed information for the different setups, the last step is to upload it into the iMOTION™ controller using the MCEDesigner. In order to do so:

1. Select Tools --> Programmer in the MCEDesigner
2. Select combined parameter files in “Program Combined File” and press start

**Figure 15** Programing combined .ldf file with the MCEDesigner.
3 Multiple Parameter Handling by iMOTION™ Solution Designer

The iMOTION™ Solution Designer (iSD) supports multiple parameter sets in one single project. It is much easier to handle multiple parameters with the iSD than the MCEWizard/Designer.

This section will cover how to configure multiple parameter sets and how to program the parameter sets to an iMOTION™ device by using the iSD.

3.1 Multiple Parameter Set Definition

3.1.1 iSD Configuration

By default, only one set of parameters is stored in FLASH and loaded into RAM of an iMOTION™ device, and the ability to select different parameter sets is disabled in Configuration Wizard in the iSD. We can enable multiple-parameter set support via the Multi-parameter input mode option in the Configuration Wizard, as shown in Figure 16.

If the multiple parameter function is not used, please select "Disable". Other options should be chosen according to the way the parameter set number was selected.

- UART: User UART
- GPIO: GPIO pins (PAR0 / PAR1 / PAR2 / PAR3)
- PARAM: Analog input pin (PARAM)

![Figure 16 Multiple Motor Parameter Set Support in Configuration Wizard iSD](image)

The Parameter Set Manager in the iSD is used to create a new parameter set. The Parameter Set Manager can be opened by selecting Project → Parameter Set Manager. Users will then see the Parameter Set Manager, as shown in Figure 17. Press "New" to create a new parameter set, input the parameter set name, and then press "Apply".
After that, the iSD generates a new parameter set and the Configuration Wizard is opened, as shown in Figure 18. Users will see the newly added parameter set name shown in the pulldown list in 'Parameter Set Configuration'.

Users can confirm the list of parameter sets in 'Project Info', which is displayed by selecting 'Project Settings' or 'Parameter Set Manager'. Figure 19 shows an example of the Project Info pane. List of parameter sets is shown in 'Parameter Sets' section. Each parameter set has an individual number, and it is used to signify that the parameter is set to be used in the MCE.
Valid parameter set IDs can range from 0 to 15 (0x00~0x0F) for FW5.x.

3.1.2 Motor ID using UART
UART command to select parameter set is same as FW1.3.x. Please refer to Section 2.1.2.

3.1.3 Motor ID using GPIO pins
As described in section 0, parameter set can be selected by using four GPIO pins (PAR0, PAR1, PAR2, and PAR3 pins).

3.1.4 Motor ID using analog input
Analog input method to select parameter set is same as FW1.3.x. Please refer to Section 2.1.4.

3.1.5 Build Project with Multiple Parameter Sets
It is necessary to build the iSD project before programming it. Please refer to [2] and [5] for how to build the project.

When the multiple parameter function is enabled, it is necessary to select the parameter set that is loaded at the device start up, as shown in Figure 20. If a script is used in the project, it is necessary to check the checkbox for script and select the script to be used in the project list.
After building the project, several files are generated in the "generated" folder of the project folder. There are two *.ldf files in the "generated" folder. One .ldf file, which contains the device name in the file name, is firmware for the specific device. Another .ldf file, "out.ldf", contains the parameter set information and the script.
The file "out.ldf" contains all of the parameter sets, the selected script, and all of the programming commands required. Table 4 below gives an example of the output file with multiple parameter sets and the script. Users should refer to [4] for greater detail. By using the iSD, users do not have to do complicated procedures like FW1.3.x, as described in chapter 2.2.

### Table 4  Example of *.ldf file generated by iMOTION™ Solution Designer

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>% iMOTION configuration file</td>
<td>file header, lines starting with '%' are comments</td>
</tr>
<tr>
<td>% iMOTIONld version: V1.1.0 - 2022-10-11</td>
<td></td>
</tr>
<tr>
<td>% Erase Script memory</td>
<td>erasing script and verification of the script</td>
</tr>
<tr>
<td>a0 22 00 05 00</td>
<td></td>
</tr>
<tr>
<td>a0 21 00 05 00</td>
<td></td>
</tr>
<tr>
<td>% linker input file</td>
<td>build information from iSD</td>
</tr>
<tr>
<td>% filename: C:...\generated\build.sdll</td>
<td></td>
</tr>
<tr>
<td>% built: 2023-03-10, 10:16:19</td>
<td></td>
</tr>
<tr>
<td>% Erase Static Parameters</td>
<td>erasing, programming and verification of system parameters</td>
</tr>
<tr>
<td>a0 22 0f 01 00</td>
<td></td>
</tr>
<tr>
<td>% Static Parameters</td>
<td></td>
</tr>
<tr>
<td>a0 20 0f 01 40 41 05 00 00 ... 00 00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>a0 20 0f 01 40 00 44 00 44 ... 00 00</td>
<td></td>
</tr>
<tr>
<td>% Verify Static Parameters</td>
<td></td>
</tr>
<tr>
<td>a0 21 0f 01 00</td>
<td></td>
</tr>
<tr>
<td>% Parameter set: myFirstParameterSet</td>
<td>erasing, programming and verification of 1st parameter set</td>
</tr>
<tr>
<td>% Erase Parameter Set</td>
<td>(parameter set number = 0, parameter set name = myFirstParameterSet)</td>
</tr>
<tr>
<td>a0 22 00 04 00</td>
<td></td>
</tr>
<tr>
<td>% Program Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 20 00 04 30 00 00 00 00 ... 00 00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>a0 20 00 04 20 02 22 00 20 ... 00 00</td>
<td></td>
</tr>
<tr>
<td>% Verify Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 21 00 04 00</td>
<td></td>
</tr>
<tr>
<td>% Parameter set: Param01</td>
<td>erasing, programming and verification of 2nd parameter set</td>
</tr>
<tr>
<td>% Erase Parameter Set</td>
<td>(parameter set number = 1, parameter set name = Param01)</td>
</tr>
<tr>
<td>a0 22 01 04 00</td>
<td></td>
</tr>
<tr>
<td>% Program Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 20 01 04 30 00 00 00 00 ... 00 00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>a0 20 01 04 30 0a 2a 73 3e ... 00 00</td>
<td></td>
</tr>
<tr>
<td>% Verify Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 21 01 04 00</td>
<td></td>
</tr>
<tr>
<td>......</td>
<td></td>
</tr>
<tr>
<td>% Parameter set: Param03</td>
<td>Parameter section continues up to the last parameter set</td>
</tr>
<tr>
<td>% Erase Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 22 03 04 00</td>
<td></td>
</tr>
<tr>
<td>% Program Parameter Set</td>
<td>erasing, programming and verification of last parameter set</td>
</tr>
<tr>
<td>a0 22 03 04 00</td>
<td></td>
</tr>
</tbody>
</table>
### Handling multiple motor/PFC parameters with iMOTION™

A guide for iMOTION™, MCEWizard, MCEDesigner and iMOTION™ Solution

**Multiple Parameter Handling by iMOTION™ Solution Designer**

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a0 20 03 04 30 00 00 00 00 ... 00 00 00 00 00</td>
<td>(parameter set number = 3, parameter set name = Param03)</td>
</tr>
<tr>
<td>... a0 20 03 04 30 0a 2a 85 3e ... 00 00</td>
<td></td>
</tr>
<tr>
<td>% Verify Parameter Set</td>
<td></td>
</tr>
<tr>
<td>a0 21 03 04 00</td>
<td></td>
</tr>
<tr>
<td>%------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>%# Build Successful - 0 Error, 0 Warning</td>
<td></td>
</tr>
<tr>
<td>%# Build Date and Time : 2023-03-10 14:49:58</td>
<td></td>
</tr>
<tr>
<td>%# Script Translator Version : 2.00.00</td>
<td></td>
</tr>
<tr>
<td>% Script Object File</td>
<td></td>
</tr>
<tr>
<td>%# SCRIPT_USER_VERSION : 001.000</td>
<td></td>
</tr>
<tr>
<td>%# Script Code Memory Size : 304 Bytes of 16 kBytes</td>
<td></td>
</tr>
<tr>
<td>% Total Number of Lines : 122</td>
<td></td>
</tr>
<tr>
<td>%# Number of Global Variable(s) : 2, Number of Flash Variable(s) : 0, Data Memory Usage : 5 Bytes of 256 Bytes</td>
<td></td>
</tr>
<tr>
<td>%# Task0 - Number of Instruction(s) : 23, Number of Variable(s) : 3, Data Memory Usage : 7 Bytes of 128 Bytes</td>
<td></td>
</tr>
<tr>
<td>%# Task1 - Number of Instruction(s) : 5, Number of Variable(s) : 0, Data Memory Usage : 0 Bytes of 128 Bytes</td>
<td></td>
</tr>
<tr>
<td>% Erase Script memory</td>
<td></td>
</tr>
<tr>
<td>a0 22 00 05 00</td>
<td></td>
</tr>
<tr>
<td>% Program Script</td>
<td></td>
</tr>
<tr>
<td>a0 20 00 05 40 65 c3 ff ff ... 56 00</td>
<td></td>
</tr>
<tr>
<td>... a0 20 00 05 30 66 01 3a 06 ... fb 1f</td>
<td></td>
</tr>
<tr>
<td>% Verify Script</td>
<td></td>
</tr>
<tr>
<td>a0 21 00 05 00</td>
<td></td>
</tr>
<tr>
<td>%------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>% Erase Script</td>
<td></td>
</tr>
<tr>
<td>a0 22 00 05 00</td>
<td></td>
</tr>
<tr>
<td>% Program Script</td>
<td></td>
</tr>
<tr>
<td>a0 20 00 05 40 65 c3 ff ff ... 56 00</td>
<td></td>
</tr>
<tr>
<td>... a0 20 00 05 30 66 01 3a 06 ... fb 1f</td>
<td></td>
</tr>
<tr>
<td>% Verify Script</td>
<td></td>
</tr>
<tr>
<td>a0 21 00 05 00</td>
<td></td>
</tr>
</tbody>
</table>

- **script code header**, lines starting with '%' are comments
- Summary of version, size and total number of lines
- Number of instructions per task, can be used for partitioning of the script task execution (see [5] and [6])
- Erasing, programming and verification of the script
4 References

[1] Getting Started with iMOTION™ 2.0
[2] Getting Started with iMOTION™ Solution Designer
[6] How to Use iMOTION™ Script Language
## Revision history

### Major changes since the last revision

<table>
<thead>
<tr>
<th>Document version</th>
<th>Date of release</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2020-12-11</td>
<td>First Release</td>
</tr>
<tr>
<td>1.1</td>
<td>2023-06-01</td>
<td>Added parameter handling with PFC function, minor wording corrections Added iMOTION™ Solution Designer information</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the type in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.