How to Use UART Interface of IMM100 and IMI110 Families

About this document

Scope and purpose

This document explains how to switch from the MCEDesigner UART protocol or iMOTION™ Solution Designer UART protocol to the User Mode UART protocol or a customized UART protocol on the UART interface of IMM100 and IMI110 series devices. It also explains how to switch from PGOUT and DIR functions to UART communication functions on the UART interface of IMI110 series devices.

Intended audience

This document is intended for those who would like to use the UART communication function on IMM100 and IMI110 series and IMD112T devices that have only one UART interface and to use PGOUT and DIR functions on IMI110 series devices.

Table of contents

About this document **1**
Table of contents **1**
1 Overview of iMOTION™ UART protocols **3**
2 IMM100 Series UART Interface Resource **4**
3 IMI110 Series UART Interface Resource **5**
4 Case 1: Switching between User Mode UART protocol and MCEDesigner UART protocol (for IMM100 series using MCEWizard / MCEDesigner) **6**
4.1 Switching from MCEDesigner UART protocol to User Mode UART protocol 6
4.2 Switching from User Mode UART protocol to MCEDesigner UART protocol 6
5 Case 2: Switching between User Mode UART protocol and iSD UART protocol (for IMM100 and IMI110 series using iSD) **8**
5.1 Switching from iSD UART protocol to User Mode UART protocol 8
5.2 Switching from User Mode UART protocol to iSD UART protocol 9
6 Case 3: Switching between iSD UART protocol and PGOUT/DIR functions (for IMI110 series using iSD) **12**
6.1 Switching from iSD UART protocol to PGOUT/DIR functions 12
6.2 Switching from PGOUT/DIR functions to iSD UART protocol 12
7 Case 4: switching between a customized UART protocol and MCEDesigner UART protocol (for IMM100 series with MCEWizard / MCEDesigner) **14**
7.1 Switching from MCEDesigner protocol to a customized UART protocol 14
7.2 Switching from a customized UART protocol to MCEDesigner UART protocol 16
7.3 Automatic switching between a customized UART protocol and MCEDesigner UART protocol 17
7.3.1 Auto-switching function design 17
7.3.2 Auto-switching function implementation 19
7.3.2.1 GPIO pin function configuration 19
7.3.2.2 Script code implementation 20

8 References 23
Revision history 24

1 Overview of iMOTION™ UART protocols

The iMOTION™ Motion Control Engine (MCE) supports four types of UART protocols:

1) MCEDesigner UART protocol

   This proprietary protocol is designed to establish communication between the MCEDesigner and iMOTION™ MCE for debugging or programming purpose. It is the default communication protocol on UART0 for iMOTION™ devices with MCE FW revisions before 5.0.

2) iMOTION™ Solution Designer (iSD) UART protocol

   This proprietary protocol is designed to establish communication between the iSD and iMOTION™ MCE for debugging or programming purposes. It is the default communication protocol on UART0 for iMOTION™ devices for MCE FW revision 5.0 and onwards.

3) User Mode UART protocol

   The User Mode UART protocol is designed to provide a simple, reliable, and scalable communication protocol for motor control applications. This allows the protocol to be easily implemented in low-end microcontrollers which work as the master to control the MCE. It supports networking (up to 15 nodes on the same network) which is required in some industrial fan or pump applications. For more details about the User Mode UART protocol please refer to the MCE Functional Reference Manual [5].

4) Customized UART protocol

   This option allows customers to implement a customized UART protocol by using the script language in the Script Engine. For more details about customized UART protocol, please refer to [6] and [7].

2 IMM100 Series UART Interface Resource

The IMM100 series devices have only one physical UART interface (RXD0 / TXD0) as shown in Figure 1. The default configuration for this UART port is to run the MCEDesigner UART protocol or iSD UART protocol for programming and debugging MCE. In practice, customers may have a need to run either User Mode UART protocol or a customized UART protocol. They may also require switching between different UART protocols to satisfy both operational and debugging requirements. In this document, examples regarding how to switch between these protocols are provided. For details regarding the IMM100 series devices, refer to [1].

Use case and script code examples provided in this document were tested on an EVAL-IMM101T-046 evaluation board based on an IMM101T-046M device using the MCEWizard (Ver. 2.3.0.1) and MCEDesigner (Ver. 2.3.0.1). For details regarding to the EVAL-IMM101T-046 board, refer to [2].

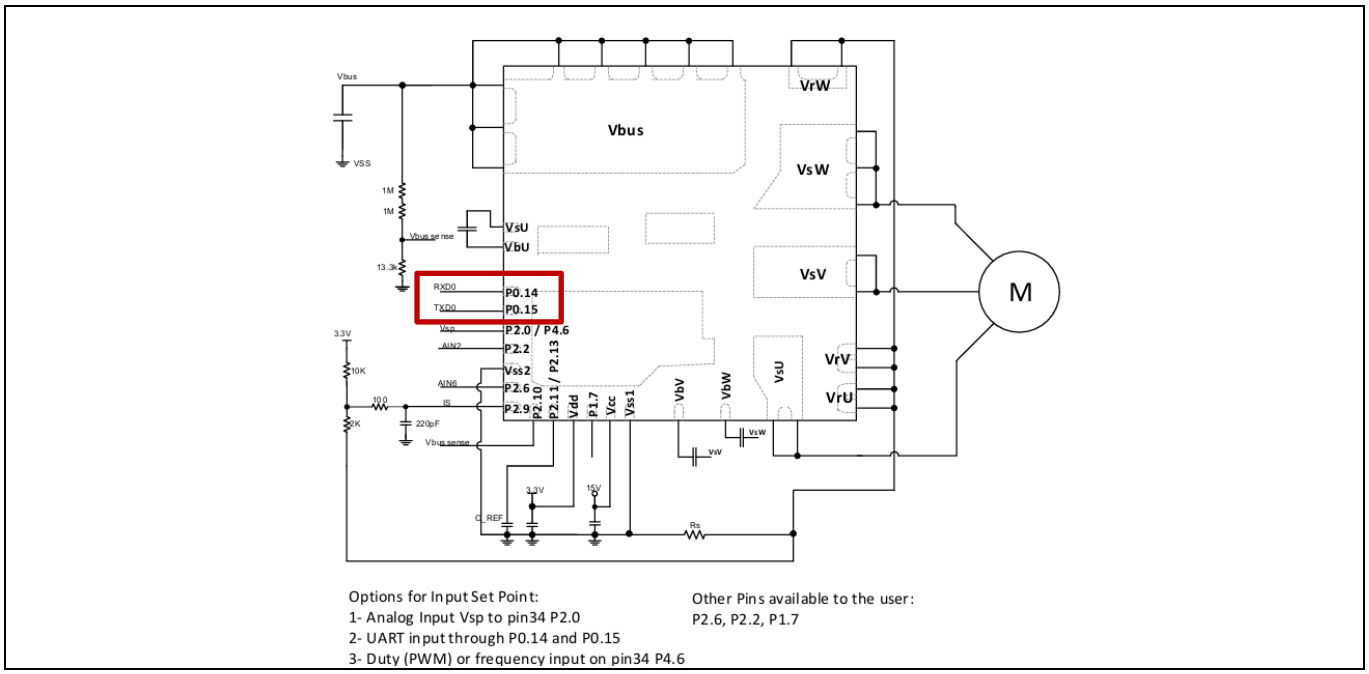


Figure 1 IMM101T UART interface pins (P0.14: RXD0; P0.15: TXD0)

3 IMI110 Series UART Interface Resource

The IMI110 series devices have only one physical UART interface (RXD0 / TXD0) as shown in Figure 2. The default configuration for this UART port is to run the iSD UART protocol for programming and debugging the MCE. In practice, customers may have a need to run either the User Mode UART protocol or a customized UART protocol. The process may also require switching between different UART protocols to satisfy both operational and debugging requirements. In this document, examples regarding how to switch between these protocols are provided. For details regarding the IMI110 series devices, refer to [3]. Because of limited pin availability, the TX0 pin is multiplexed with PGOUT function, and the RX0 pin is multiplexed with DIR function. Examples regarding how to switch between the UART protocol and PGOUT/DIR functions are provided as well.

Use case examples provided in this document were tested on an EVAL-IMI111T-026 evaluation board based on IMI111T-026H device using iSD (Ver. 1.x). For details regarding to the EVAL-IMI111T-026 board, refer to [4].

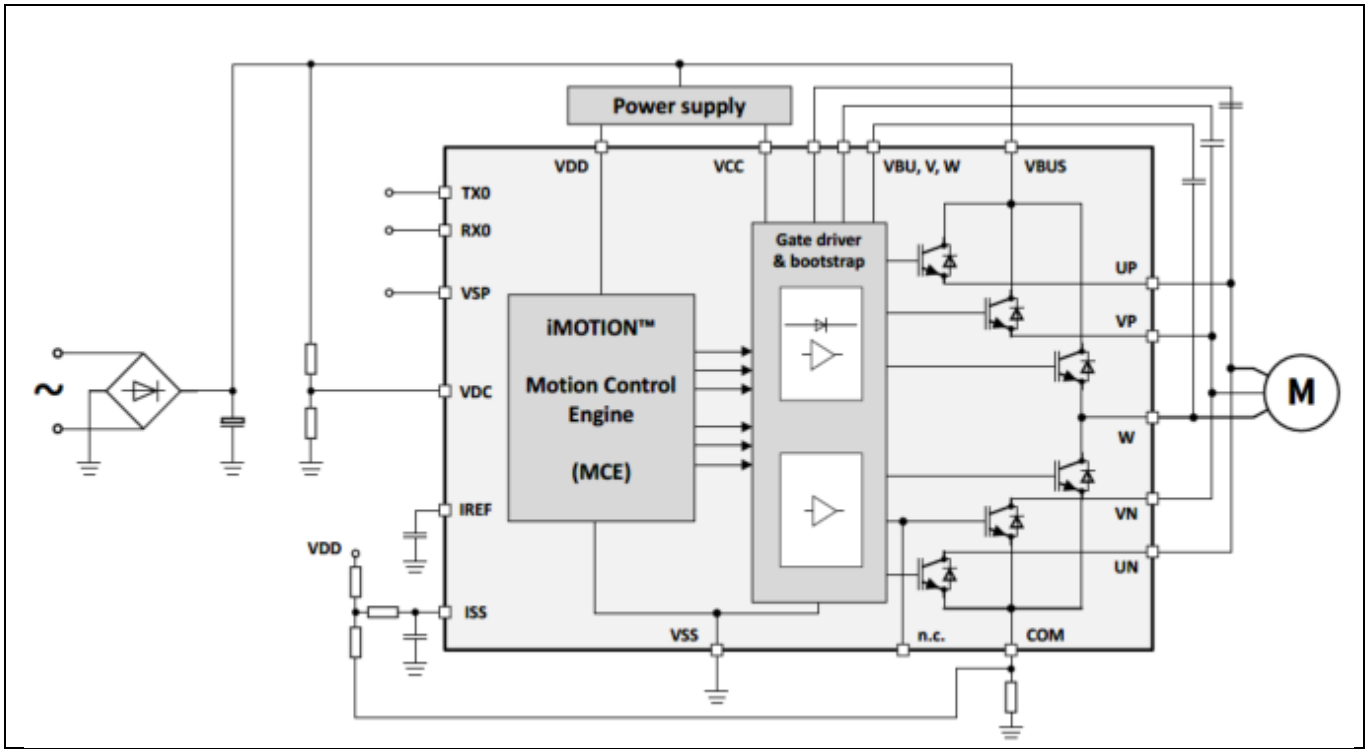


Figure 2 IMI111T UART interface pins (RXD0/DIR; TXD0/PGOUT)

4 Case 1: Switching between User Mode UART protocol and MCEDesigner UART protocol (for IMM100 series using MCEWizard / MCEDesigner)

4.1 Switching from MCEDesigner UART protocol to User Mode UART protocol

Step 1: As shown in Figure 3, open the MCEWizard and select “Advanced Mode” from the top pull-down list and go to the “System” tab. Select “UART0” from the pull-down list for Question 19 “User UART Function Definition”. Then generate a parameter file accordingly following the procedure mentioned in [2].

Step 2: Use the MCEDesigner to download the generated parameter file.

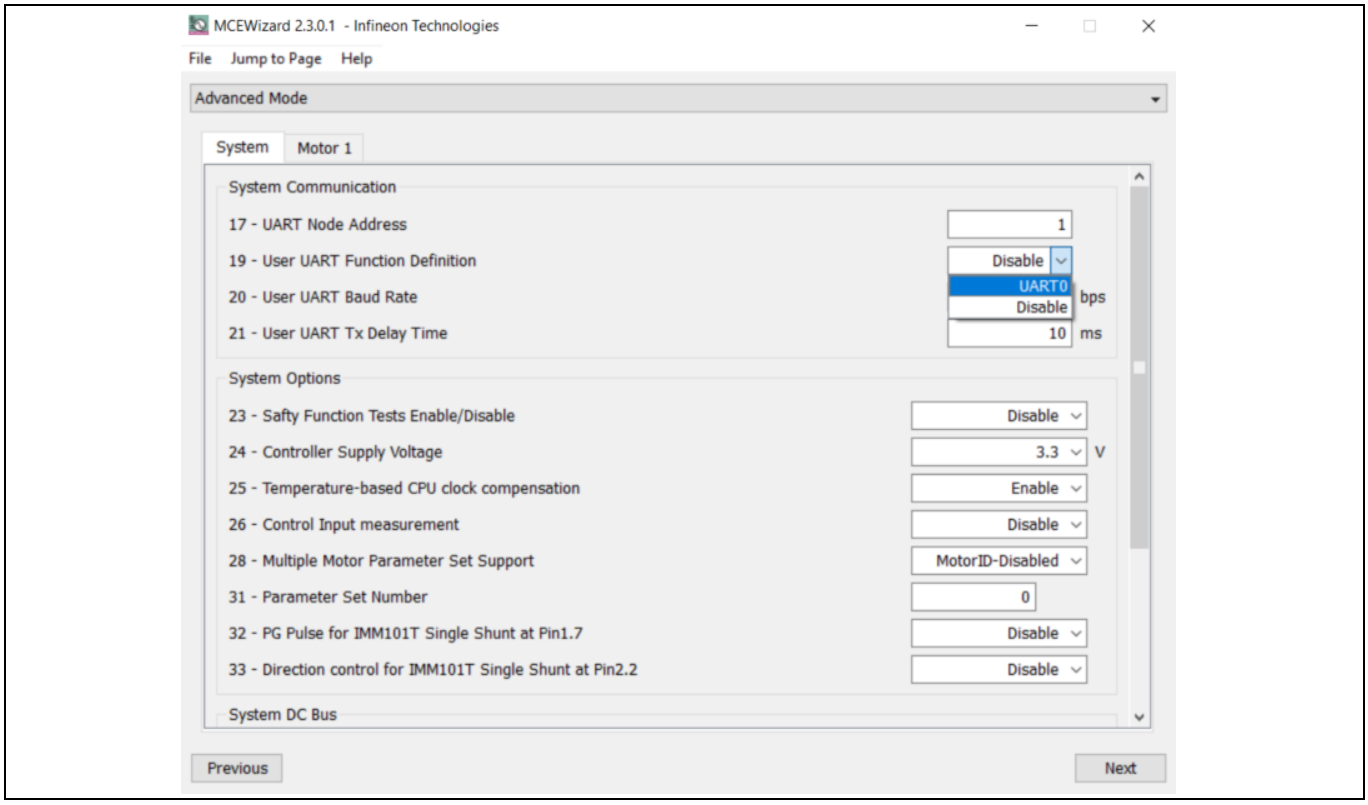


Figure 3 User Mode UART protocol configuration in MCEWizard

Once the newly generated parameter file is downloaded to the target successfully, the MCEDesigner will appear to lose communication with the target. This is normal, as the MCE is now hosting the User Mode UART protocol in the UART0 interface, which is not recognized by the MCEDesigner.

4.2 Switching from User Mode UART protocol to MCEDesigner UART protocol

In order to switch from the User Mode UART protocol to the MCEDesigner UART protocol, users need to change the parameter file by disabling the User Mode UART function.

Step 1: Select “Disable” from the pull-down list for Question 19 “User UART Function Definition”, as shown in Figure 3. Then generate a parameter file following the procedure mentioned in [2].

Step 2: Open the MCEDesigner and the relevant IMM101T irc file.

Step 3: Click the “Cancel” button on the “Communication Error” window as shown in Figure 4.

Step 4: Select “Tools” in the menu, click “Programmer”, and load the generated parameter file with the User Mode UART function disabled as shown in Figure 5. Do not click “Start” at this moment.

Step 5: First, power off the IMM101T board. Then click the “Start” button in Figure 5 and wait two seconds before powering on the IMM101T board again and the programming process will start. Once this step goes through successfully, the UART0 will run the MCEDesigner UART protocol and the communication between the MCE and MCEDesigner will be restored, which is indicated by a green light in the status bar of the MCEDesigner. If this step fails, please retry the process from Step 3.

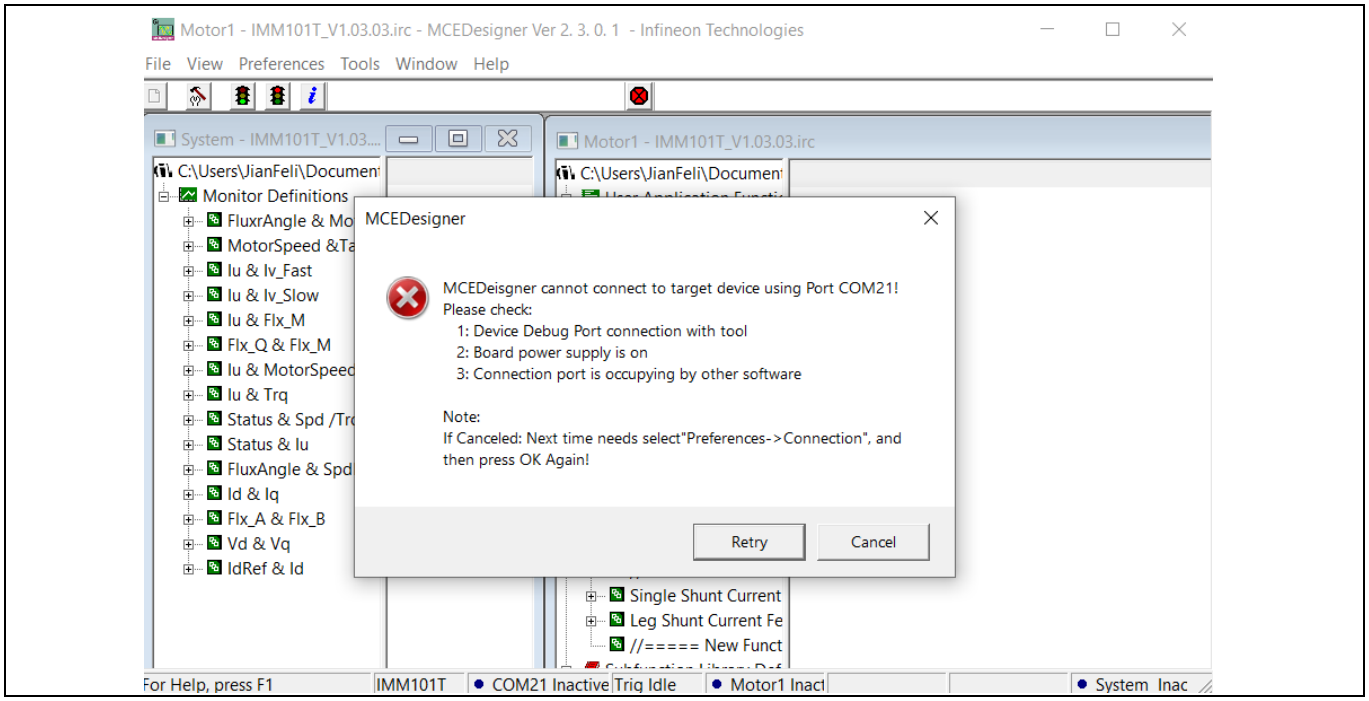


Figure 4 “Communication error” window in MCEDesigner

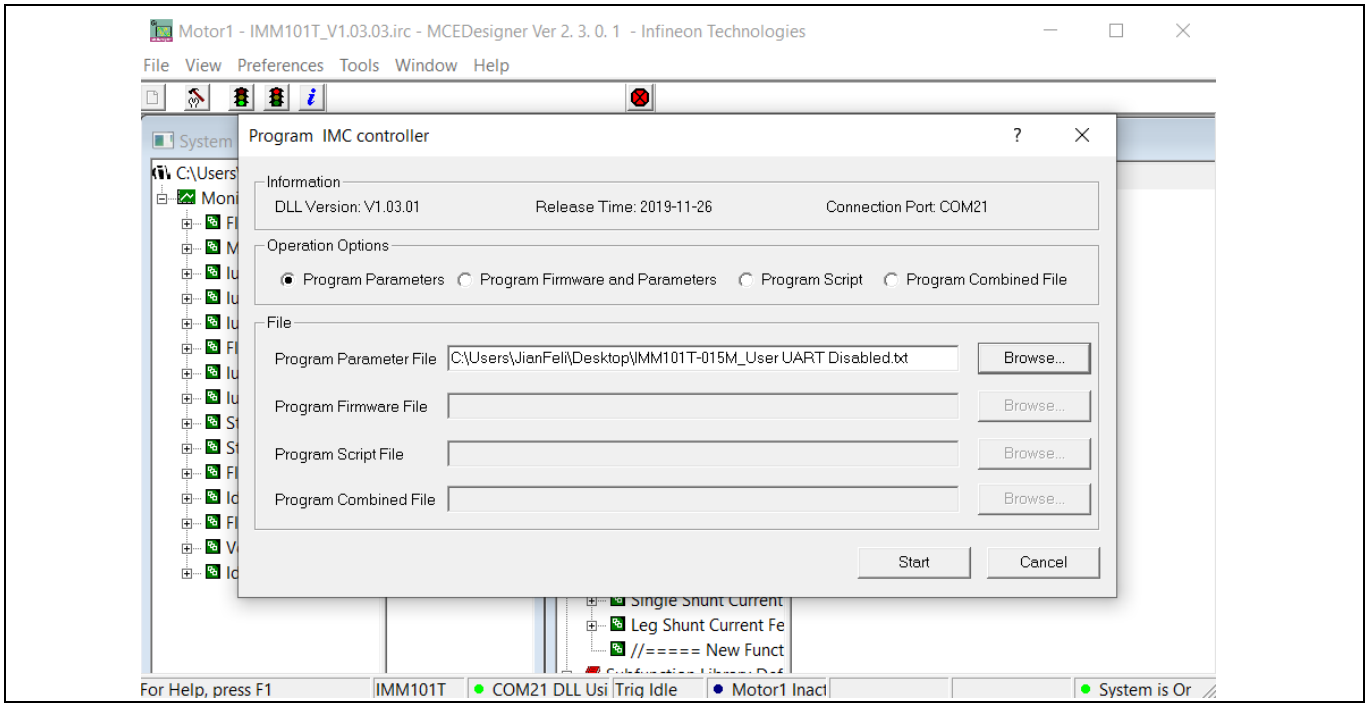


Figure 5 “Ready to program parameter file” window in MCEDesigner

5 Case 2: Switching between User Mode UART protocol and iSD UART protocol (for IMM100 and IMI110 series using iSD)

5.1 Switching from iSD UART protocol to User Mode UART protocol

Step 1: As shown in Figure 6, open the Configuration Wizard and select “IC Configuration” from the Global Configuration and go to “COMM” section. Select “UART0” from the pull-down list for “User/Host COMM port selection”.

Step 2: as shown in Figure 7, this enables communication with an external user host MCU. The selected UART will be configured with the baud rate selected and the communication will follow the ‘user mode protocol’ as described in the MCE Functional Reference Manual [5].

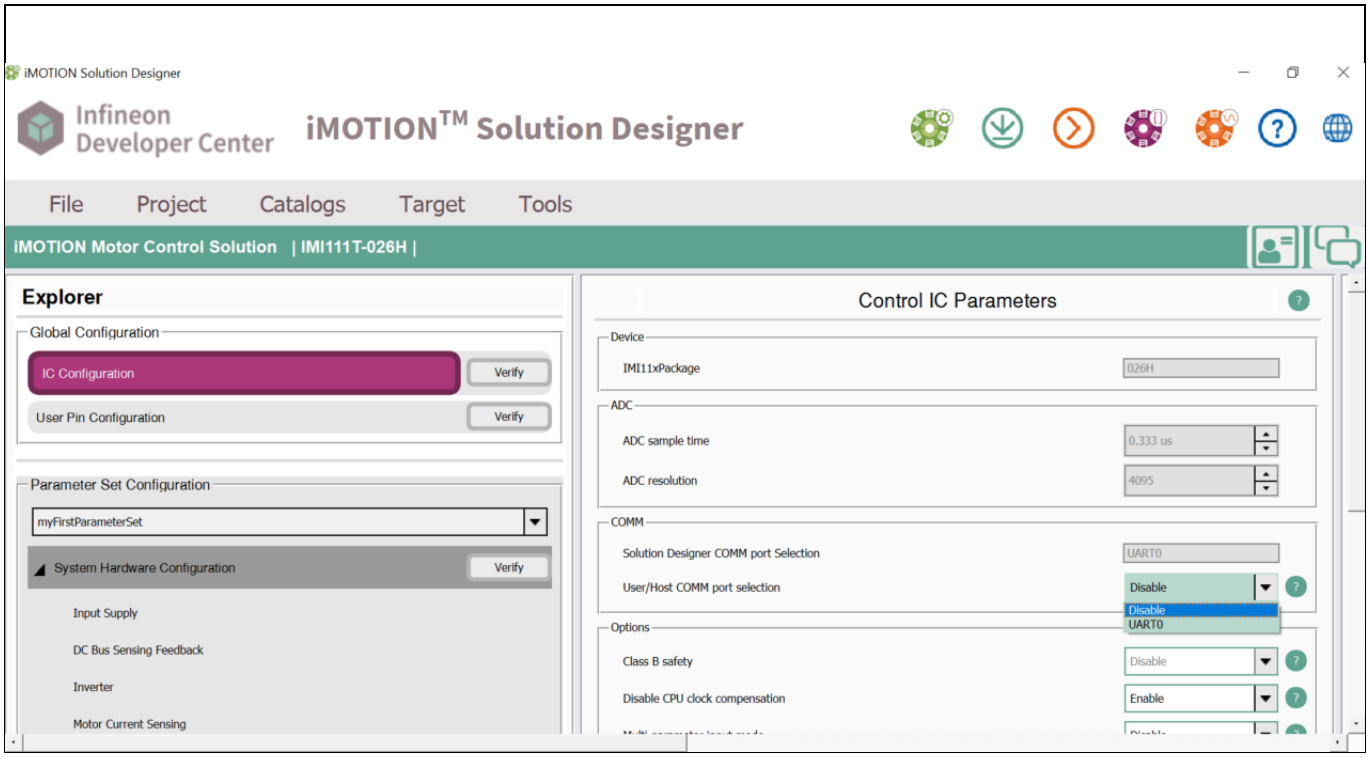


Figure 6 User Mode UART protocol configuration in iSD

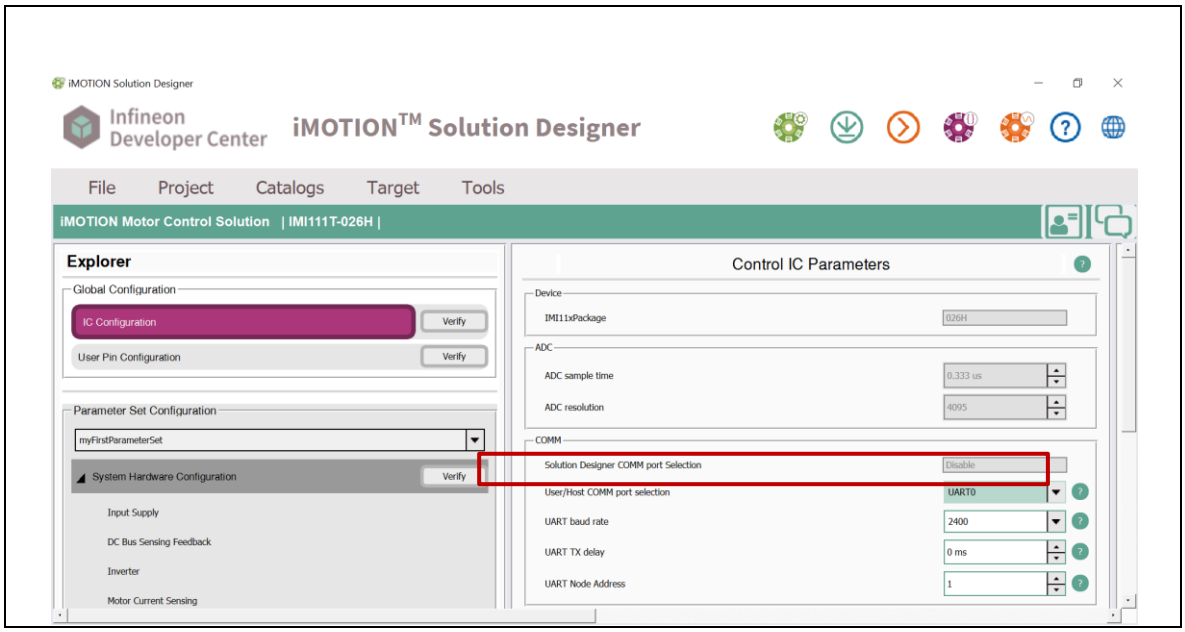


Figure 7 User Mode UART protocol configuration in iSD

Once the new configuration is built and downloaded to the target successfully, the iSD will appear to lose communication with the target. This is normal, as the MCE is now hosting the User Mode UART protocol in the UART0 interface, which is not recognized by the iSD.

5.2 Switching from User Mode UART protocol to iSD UART protocol

In order to switch from the User Mode UART protocol to the iSD UART protocol, users need to reverse the selections in section 5.1 by disabling the User Mode UART.

Step 1: Select “Disable” from the pull-down list for “User/Host COMM port selection”, as shown in Figure 6.

Step 2: Verify the configuration and build the project.

Step 3: Select “Tools” in the menu and click the “Programmer” option to launch the Programmer tool. Click the “Connect” button as shown in Figure 8.

Step 4: A pop-up “Connect” window will show up as shown in Figure 9. Choose the appropriate COM port in the drop-down list that matches the JLink debugger being used, then click the “Connect” button below. At this point the Programmer window will show the message “Target is not connected, please check cables, power supply, and target interface settings.” as shown in Figure 10. This is normal, as the MCE had been configured to run the User Mode UART protocol in the UART0 interface, which is not recognized by the iSD.

Step 5: Power off the IMM101T/IMI111T board at this time. Then click the “Connect” button in the Programmer window as shown in Figure 10, and then power on the board within 10 seconds. At this point, the target will be connected to the iSD at power-on. The iSD will read and show the connected device information, boot mode, FW version, and the FW CRC value in the Programmer window as shown in Figure 11.

Step 6: Click the “Program” button to initialize the programming process as shown in Figure 11. When the programming process goes through successfully, the Programmer window will show the message “Target was programmed successfully”, as shown in Figure 12. Click the “Close” button to close the Programmer window. At this point, the UART0 will run the iSD UART protocol and the communication between the MCE and iSD will be restored. If this step fails, please click the “Close” button and retry from Step 3.

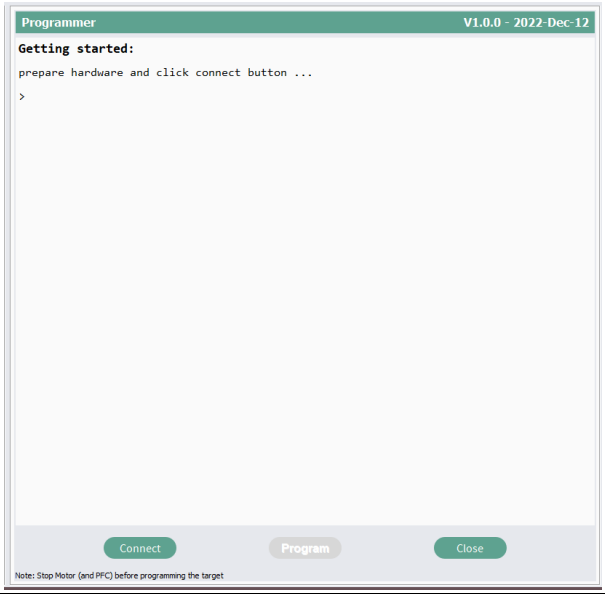


Figure 8 “Programmer” window in iSD: start-up

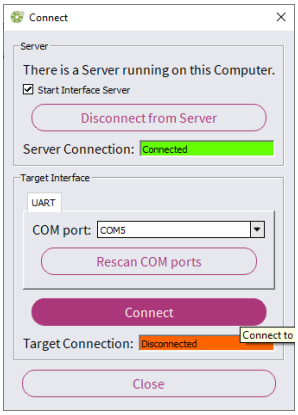


Figure 9 “Connect” pop-up window



Figure 10 “Programmer” window in iSD: target is not connected

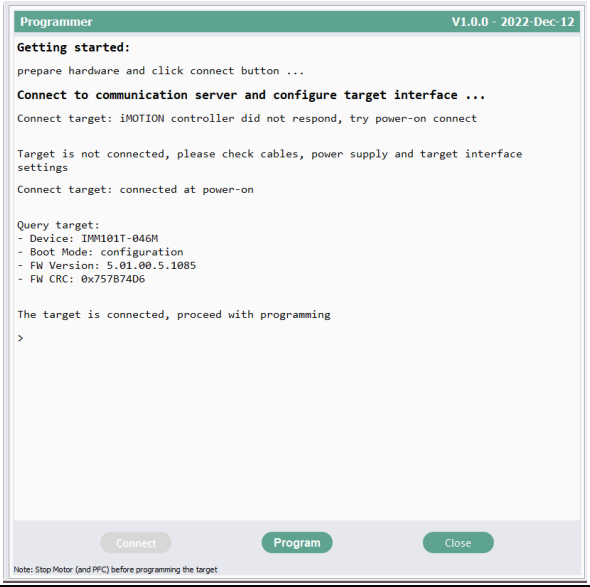


Figure 11 “Programmer” window in iSD: target connected at power-on

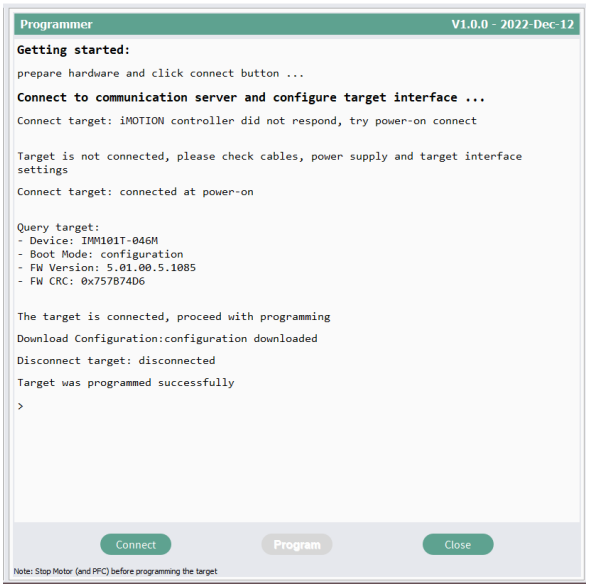


Figure 12 “Programmer” window in iSD: target programmed successfully

6 Case 3: Switching between iSD UART protocol and PGOUT/DIR functions (for IMI110 series using iSD)

6.1 Switching from iSD UART protocol to PGOUT/DIR functions

Step 1: As shown in Figure 13, open the Configuration Wizard and select “Application” from the Motor Control Configuration and go to the “Summary” section. Select “Enable” from the pull-down list for “PG output”, and select “VSP” from the pull-down list for “Command input”.

Step 2: This enables the multiplexing functions of PGOUT and DIR functions for IMI111T. The TX0 is disabled and PGOUT is enabled while RX0 is disabled and DIR is enabled.

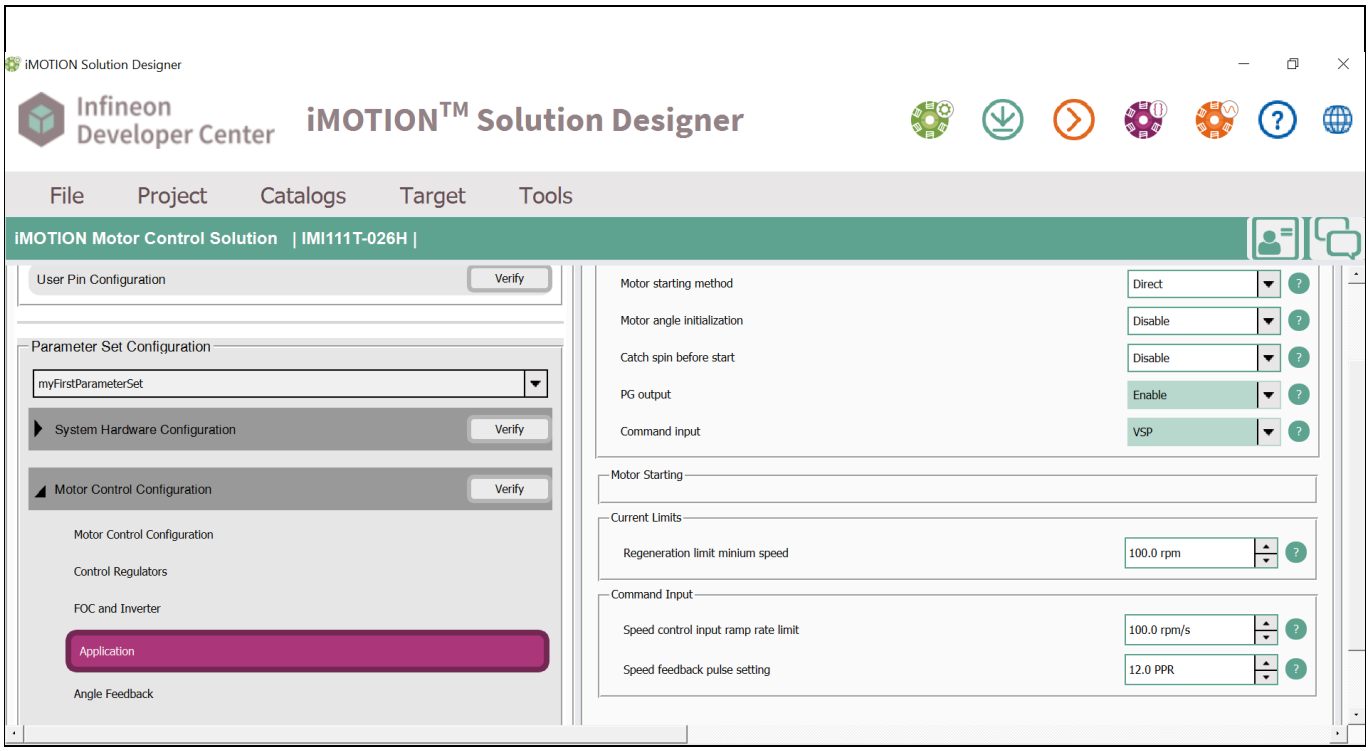


Figure 13 PGOUT/DIR configuration in iSD for IMI111T

Once the new configuration is built and downloaded to the target successfully, the iSD will appear to lose communication with the target. This is normal, as the MCE is now hosting the PGOUT and DIR functions in the UART0 interface, which is not recognized by the iSD.

6.2 Switching from PGOUT/DIR functions to iSD UART protocol

In order to switch from the PGOUT/DIR functions to the iSD UART protocol, one needs to reverse the selections in section 6.1 by disabling the PGOUT/DIR functions.

Step 1: Select “Disable” from the pull-down list for “PG output”, and select “UART” from the pull-down list for “Command input”.

Step 2: Verify the configuration and build the project.

Step 3: Select “Tools” in the menu and click the “Programmer” option to launch the Programmer tool. Click the “Connect” button as shown in Figure 8.

Step 4: A pop-up “Connect” window will appear as shown in Figure 9. Choose the appropriate COM port in the drop-down list that matches the JLink debugger being used, then click the “Connect” button below. At this point the Programmer window will show the message “Target is not connected, please check cables, power supply, and target interface settings.” as shown in Figure 10. This is normal, as the MCE had been configured to run the User Mode UART protocol in the UART0 interface, which is not recognized by the iSD.

Step 5: Power off the IMI111T board at this time. Then click the “Connect” button in the Programmer window as shown in Figure 10, then power on the board within 10 seconds. At this point, the target will be connected to the iSD at power-on. The iSD will read and show the connected device information, boot mode, FW version, and the FW CRC value in the Programmer window as shown in Figure 11.

Step 6: Click the “Program” button to initialize the programming process as shown in Figure 11. When the programming process goes through successfully, the Programmer window will show the message “Target was programmed successfully.” as shown in Figure 12. Click the “Close” button to close the Programmer window. At this point, the UART0 will run the iSD UART protocol, and the communication between the MCE and iSD will be restored. If this step fails, please click the “Close” button and retry from Step 3.

7 Case 4: switching between a customized UART protocol and MCEDesigner UART protocol (for IMM100 series with MCEWizard / MCEDesigner)

7.1 Switching from MCEDesigner protocol to a customized UART protocol

To switch from the MCEDesigner protocol to a customized UART protocol, users need to configure the UART0 by using script language.

For more details about UART protocol customization, please refer to [7].

For more details about script language, please refer to [6].

Assume a simple customized UART protocol that contains 5 bytes per frame including a checksum byte at the end. The frame structure details are shown in Table 1 below.

Table 1 Host-to-Slave Frame Structure

Head Byte	Command Byte	Data Byte 1	Data Byte 2	Checksum Byte
0x0E	XX	XX	XX	XX

The communication is always initiated by the host while the MCE device serves as the slave. When the Command Byte = 0x03, it is interpreted as setting target speed by using the value of the following two Data Bytes (TargetSpeed = Data Byte 1 + Data Byte 2 x 256). The slave replies with a frame whose structure details are shown in Table 2 below. If the checksum checks out, then the slave replies with a frame whose Command Byte = 0x13. If the checksum fails, then the slave replies with a frame whose Command Byte = 0xEE.

Table 2 Slave-to-Host Frame Structure

Head Byte	Command Byte	Data Byte 1	Data Byte 2	Checksum Byte
0x11	XX	lower 8 bit of “status” word	higher 8 bit of “status” word	XX

The values of the frame parameters in this example are listed in Table 3 below.

Table 3

Parameter	Value
Baudrate	115200 bit/s
Parity	none
StopBits	1 bit
HalfDuplex	half duplex
RxTimeout	500 ms
TxDelay	0 ms
RxFlag	0x0E
TxFlag	0x11
RxDataLength	4
TxDataLength	4
Full data frame length	5

Code Listing 1 shows one example of script code implementation to realize this simple customized UART protocol. In this example, two predefined methods are used to configure the UART0 to Buffer mode: `UART_DriverInit()` and `UART_BufferInit()`. These methods need to be executed only once in Script Engine. So they will be called in `Script_Task0_init()` or `Script_Task1_init()`. For detailed description of the parameters of the functions `UART_DriverInit()` and `UART_BufferInit()`, please refer to the MCE Functional Reference Manual [5].

Code Listing 1 Script code example for a customized UART protocol

```
#SET SCRIPT_USER_VERSION (1.00)  /*Script version value should be 255.255*/
#SET SCRIPT_TASK0_EXECUTION_PERIOD    (5)  /*Script execution time for Task0 in mS, maximum value 65535*/
#SET SCRIPT_TASK1_EXECUTION_PERIOD    (1)  /*Script execution time for Task1 in 10mS, maximum value 65535*/
#SET SCRIPT_START_COMMAND (0x3)  /* Start command, Task0 : Bit0, Task1 : Bit1; if bit is set, script executes after init */
#SET SCRIPT_TASK0_EXECUTION_STEP (15)  /* Script Task0  step, This defines number of lines to be executed every 1mS*/
#SET SCRIPT_TASK1_EXECUTION_STEP (150) /* Script Task1  step, This defines number of lines to be executed every 10mS*/

int status;
int databits;


Script_Task0_init()
{

}

Script_Task0()
{


}
Script_Task1_init()
{
        status = 0;
        databits = 0;
        UART_DriverInit(0 /*channel*/, 0 /*rxInvert*/, 0 /*txInvert*/, 115200 /*baudrate*/, 8 /*dataBits*/, 0 /*parity*/, 1 /*stopBits*/);
        UART_BufferInit(1 /*halfDuplex*/, 500 /*rxTimeout*/, 2 /*txDelay*/, 0 /*txByteDelay*/, 0x0E /*rxFlag*/, 0x11 /*txFlag*/, 4 /*rxDataLength*/, 4 /*txDataLength*/);
}

Script_Task1()
{
        int checksum;

        status= UART_GetStatus();

        if(UART_GetStatus() & 0x0100)/*UART_IsRxBufferFull*/
```

Code Listing 1 Script code example for a customized UART protocol

```
037         {
038                 checksum = (-(0x0E + UART_RxBuffer(0) + UART_RxBuffer(1)
    + UART_RxBuffer(2))) & 0xFF; /* 8bit checksum */
039
040                 if(checksum == (0xFF & (UART_RxBuffer(3))))
041                 {
042                       /* process command */
043                       /*Set Motor Speed Command */
044                       if(0x03 == UART_RxBuffer(0))
045                       {
046                             TargetSpeed = UART_RxBuffer(1) +
    (UART_RxBuffer(2) << 8);
047
048                             /* prepare response */
049                             UART_TxBuffer(0, 0x13);
050                             UART_TxBuffer(1, status & 0xff);
051                             UART_TxBuffer(2, status >> 8);
052                             UART_TxBuffer(3, -(0x11 + 0x13 + (status &
    0xff) + (status >> 8)) /* no truncation required */);
053                       }
054                 }
055                 else
056                 {
057                       UART_TxBuffer(0, 0xEE);
058                       UART_TxBuffer(1, status & 0xff);
059                       UART_TxBuffer(2, status >> 8);
060                       UART_TxBuffer(3, -(0x11 + 0xEE + (status & 0xff)
    + (status >> 8)) /* no truncation to 8bit required */ );
061                 }
062                 UART_Control(0x0500 /* UART_SendTxBuffer |
    UART_ClrRxBufferFlag */);
063         }
064 }
```

Follow these steps to switch from the MCEDesigner protocol to the customized UART protocol.

Step 1: Write script code to initialize the UART0 using the predefined methods per the customized UART protocol requirement.

Step 2: Compile the script code using the MCEWizard. Download the generated script file to the IMM101T device by using the MCEDesigner. Then the UART0 on IMM101T will be executing the customized UART protocol.

It is normal to observe that the communication between the MCE and MCEDesigner is lost after the script is successfully downloaded to the device. This is because the MCE is now hosting the customized UART protocol on UART0 interface, which is not recognized by the MCEDesigner.

7.2 Switching from a customized UART protocol to MCEDesigner UART protocol

Switching from a customized UART protocol implemented in script to the MCEDesigner UART protocol requires disabling the customized UART protocol implementation in the script file. Please operate following the steps below:

Step 1: Open the MCEDesigner and the relevant IMM101T irc file.

Step 2: Click the “Cancel” button on the “communication error” window as shown in Figure 4.

Step 3: Click “Tools” and “Programmer” from the menu so that users can download the revised script file that has disabled the customized UART protocol implementation.

Step 4: Power off the IMM101T board first. Click the “Start” button in the “Program IMC controller” window as shown in Figure 14. Wait two seconds before powering on IMM101T board again and the programming process will start. If this step goes through successfully, then the UART0 will run the MCEDesigner UART protocol and the communication between the MCE and MCEDesigner will be restored. This is indicated by a green light in the status bar of the MCEDesigner. If this step fails, please retry from Step 2.

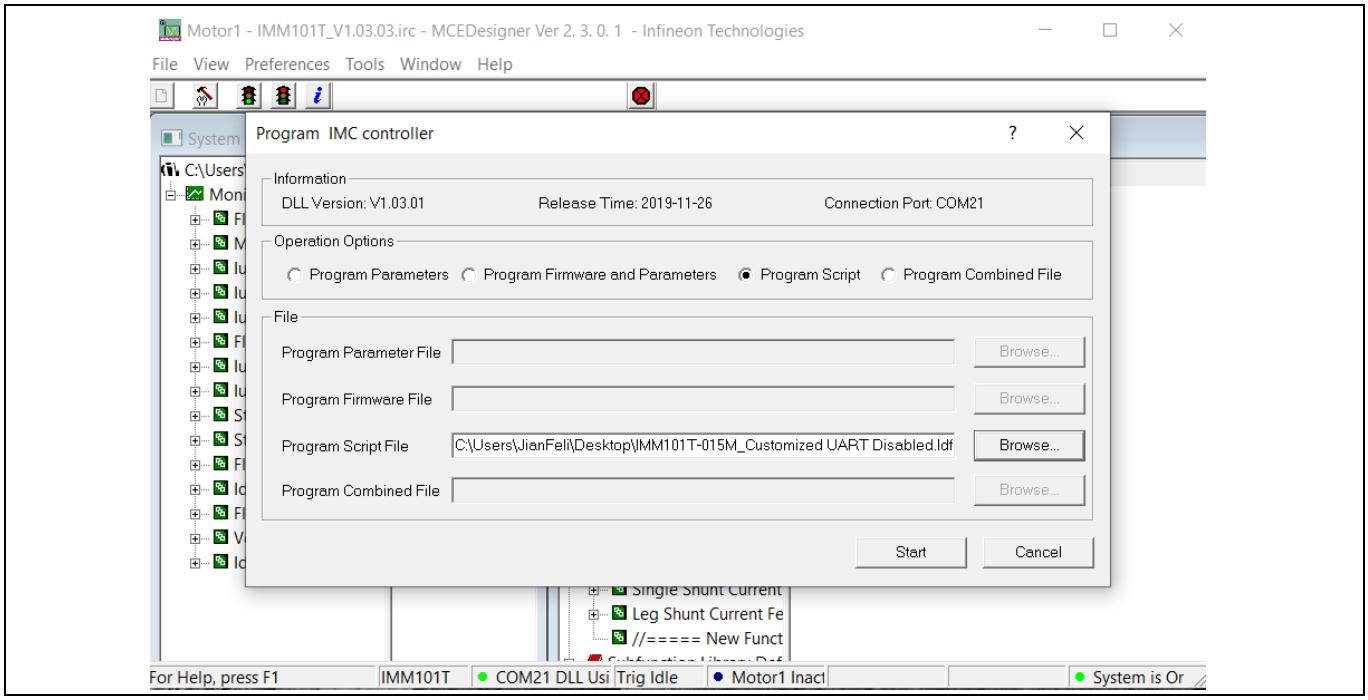


Figure 14 “Ready to program script file” window in MCEDesigner

7.3 Automatic switching between a customized UART protocol and MCEDesigner UART protocol

This section introduces a method to switch from a customized UART protocol to the MCEDesigner UART protocol automatically.

7.3.1 Auto-switching function design

Automatic switching between a customized UART protocol and the MCE Designer UART protocol can be achieved by using a spare GPIO pin and scripting. In this example, we choose GPIO5 (P2.6) for this purpose and configure it as digital input in MCEWizard. By default, when it is configured as digital input, the internal pull-down function is enabled to ensure that the pin is in logic-low level when it is left floating. As shown in Figure 15, this GPIO5 pin is connected with a switch so that users can toggle the switch to set the pin to logic high (3.3V) or logic low level (GND).

Upon powering on the device, the script code reads the specified GPIO5 pin status first. Assume a user starts off by turning the switch on, then the input value of the GPIO5 pin is read as logic high. In that case, the script initializes the UART0 interface to host a customized UART protocol. Otherwise, the script leaves the UART0 in the initial state as configured. Thus, the UART0 would still host the MCEDesigner UART protocol as configured in the parameter file. If a user intends to switch from running a customized UART protocol to the MCEDesigner UART protocol, they will need to turn off the switch to connect the GPIO5 pin to logic low level through an internal pull-down resistor of the MCE. The power to the device needs to be cycled to reconfigure it to running the MCEDesigner UART protocol.

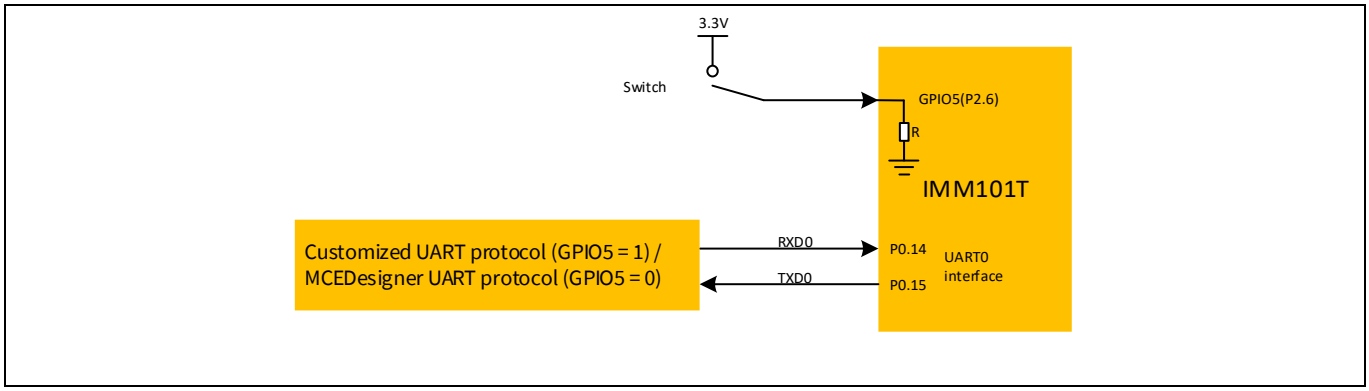


Figure 15 Auto-switching algorithm diagram

Figure 16 shows an example of the auto-switching script code flow chart. The customized UART protocol is enabled when the GPIO5 pin is set to logic high level (GPIO5_IN = 1). Since the operation of reading the GPIO5 input value is implemented in both the Script_task1_init() as well as the Script_task1() function, switching to the customized UART is possible either when the device starts or during run time. The MCEDesigner UART protocol is used when GPIO5 pin is set to logic low level (GPIO5_IN = 0) on starting the device.

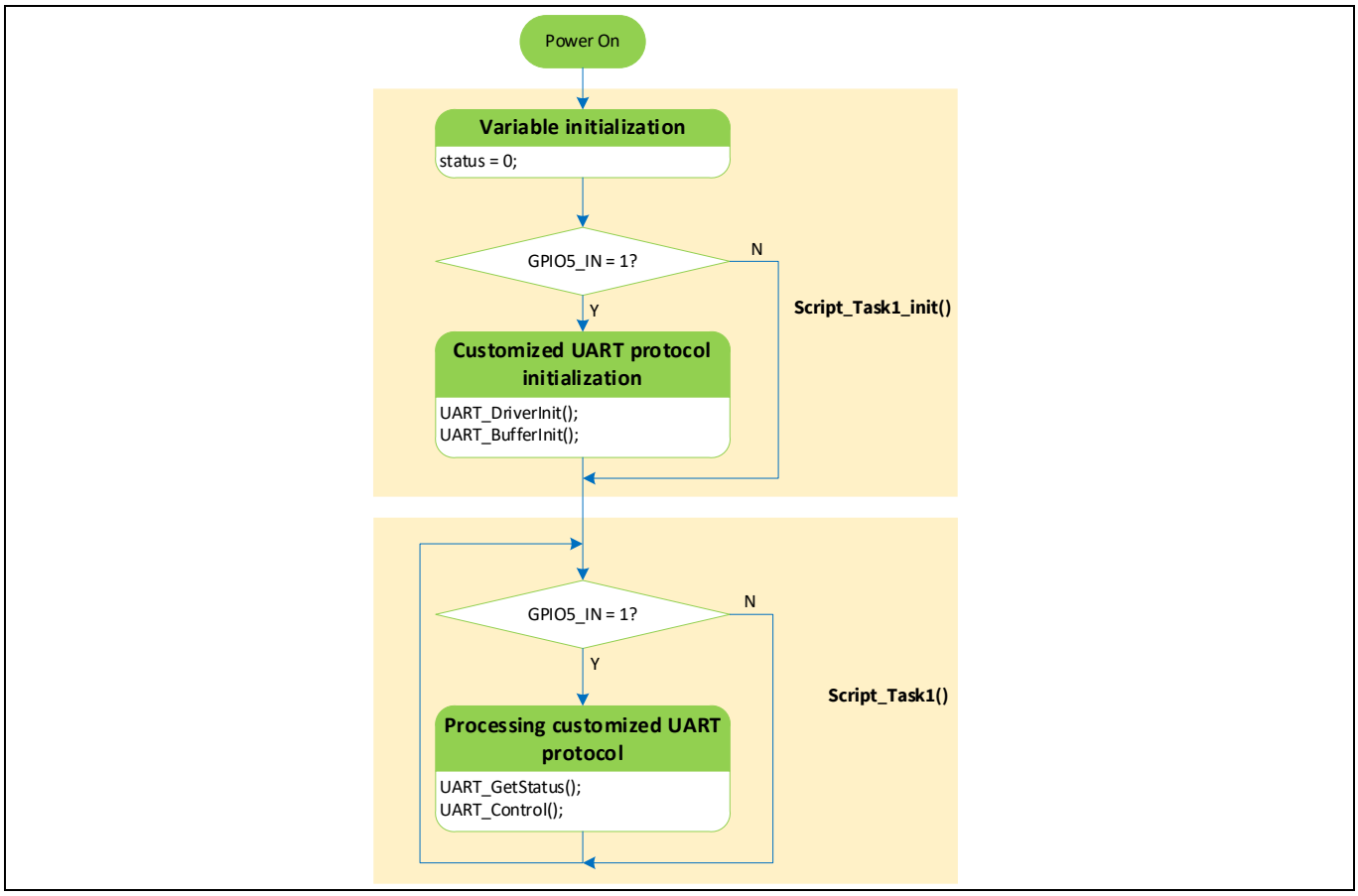


Figure 16 Auto-switching script code flow chart

7.3.2 Auto-switching function implementation

7.3.2.1 GPIO pin function configuration

Before implementing script code, the GPIO5 (P2.6) pin needs to be set as digital input function as shown in Figure 17. This allows its value to be properly read from scripting.

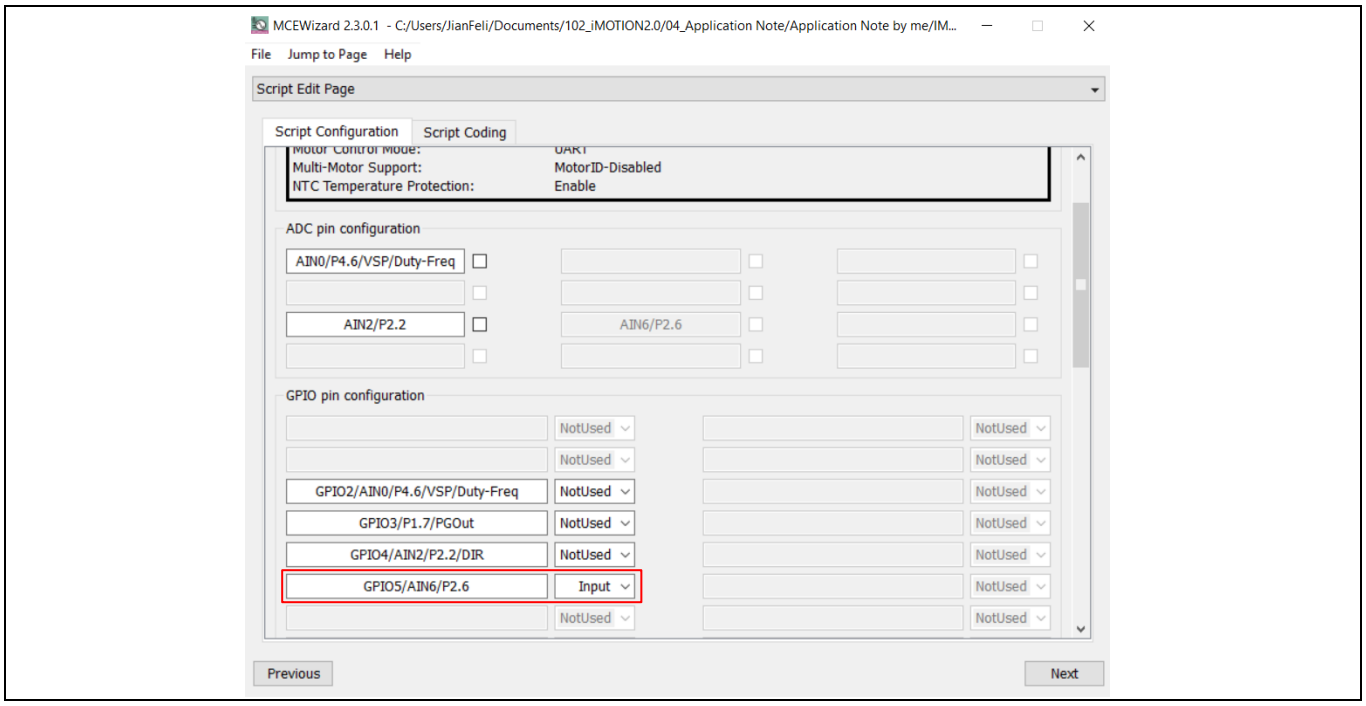


Figure 17 Script Configuration for GPIO5 pin function

7.3.2.2 Script code implementation

Code Listing 2 shows an example of script code implementation, realign this function of automatic switching between a customized UART protocol and MCEDesiger UART protocol running on UART0. The customized UART protocol mentioned in this example is the same one mentioned in Section 7.1.

Code Listing 2 Script code example for auto-switching function between a customized UART protocol and MCEDesiger UART protocol

```
#SET SCRIPT_USER_VERSION (1.00)  /*Script version value should be
255.255*/
#SET SCRIPT_TASK0_EXECUTION_PERIOD    (5)  /*Script execution time for
Task0 in mS, maximum value 65535*/
#SET SCRIPT_TASK1_EXECUTION_PERIOD    (1)  /*Script execution time for
Task1 in 10mS, maximum value 65535*/
#SET SCRIPT_START_COMMAND (0x3)  /* Start command, Task0 : Bit0, Task1
: Bit1; if bit is set, script executes after init */
#SET SCRIPT_TASK0_EXECUTION_STEP (15)  /* Script Task0  step, This
defines number of lines to be executed every 1mS*/
#SET SCRIPT_TASK1_EXECUTION_STEP (150) /* Script Task1  step, This
defines number of lines to be executed every 10mS*/

int status;
int databits;


Script_Task0_init()
{

}

Script_Task0()
```

Code Listing 2 **Script code example for auto-switching function between a customized UART protocol and MCEDesiger UART protocol**

```
018 {
019
020
021 }
022 Script_Task1_init()
023 {
024         status = 0;
025         databits = 0;
026         if(GPIO5_IN == 1)
027         {
028                 UART_DriverInit(0 /*channel*/, 0 /*rxInvert*/, 0
    /*txInvert*/, 115200 /*baudrate*/, 8 /*dataBits*/, 0 /*parity*/, 1
    /*stopBits*/);
029                 UART_BufferInit(1 /*halfDuplex*/, 500 /*rxTimeout*/, 2
    /*txDelay*/,  0 /*txByteDelay*/, 0x0E /*rxFlag*/, 0x11 /*txFlag*/, 4
    /*rxDataLength*/, 4 /*txDataLength*/);
030         }
031 }
032
033 Script_Task1()
034 {
035         int checksum;
036
037         if(GPIO5_IN == 1)
038         {
039
040                 status= UART_GetStatus();
041
042                 if(UART_GetStatus() & 0x0100)/*UART_IsRxBufferFull*/
043                 {
044                         checksum = (-(0x0E + UART_RxBuffer(0) +
    UART_RxBuffer(1) + UART_RxBuffer(2))) & 0xFF; /* 8bit checksum */
045
046                         if( checksum == (0xFF & (UART_RxBuffer(3))))
047                         {
048                                 /* process command */
049                                 /*Set Motor Speed Command */
050                                 if( 0x03 == UART_RxBuffer(0) )
051                                 {
052                                         TargetSpeed = UART_RxBuffer(1) +
    (UART_RxBuffer(2) << 8);
053
054                                         /* prepare response */
055                                         UART_TxBuffer(0, 0x13);
056                                         UART_TxBuffer(1, status & 0xff);
057                                         UART_TxBuffer(2, status >> 8);
058                                         UART_TxBuffer(3, -(0x11 + 0x13 +
    (status & 0xff) + (status >> 8)) /* no truncation required */);
059                                 }
060                         }
061                         else
062                         {
063                                 UART_TxBuffer(0, 0xEE);
```

Code Listing 2 **Script code example for auto-switching function between a customized UART protocol and MCEDesiger UART protocol**

```
064                                 UART_TxBuffer(1, status & 0xff);
065                                 UART_TxBuffer(2, status >> 8);
066                                 UART_TxBuffer(3, -(0x11 + 0xEE + (status &
   0xff) + (status >> 8)) /* no truncation to 8bit required */);
067                         }
068                         UART_Control(0x0500 /* UART_SendTxBuffer |
   UART_ClrRxBufferFlag */);
069                 }
070         }
071 }
```

8 References

[1] IMM100 series - iMOTION™ IPM for motor control Data Sheet (REV 1.2).

[2] EVAL-IMM101T User Manual (REV 1.2).

[3] IMI111 - iMOTION™ IPM for motor control Datasheet (REV 1.1)

[4] EVAL-IMI111T User Manual (REV 1.2)

[5] iMOTION™ Motor Control Engine Functional Reference Manual (REV 1.1).

[6] How to use iMOTIOM™ script language (REV 1.1).

[7] How to use iMOTION™ Configurable UART (REV 1.0)

Revision history

Document version	Date of release	Description of changes
1.0	3/29/2023	Initial release.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-03-29

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN2022-07

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 types 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.