MGC3140/MXG3141 GestIC® Library Interface Description User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Trademarks

ISBN: 978-1-5224-7820-1

Table of Contents

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This document applies exclusively to MGC3140 and MXG3141. If not differently stated, "GestIC®" is used to refer to both devices.

This chapter contains general information that will be useful to know before using the MGC3140/MXG3141 GestIC® Library Interface. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- Recommended Reading
- The Microchip Website
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes the MGC3140/MXG3141 GestIC Library and is organized as follows:

- **Chapter 1. Introduction**
- **Chapter 2. GestIC® Host Interface**
- **Chapter 3. GestIC® Library Message Interface**
- **Chapter 4. GestIC® Library Message Reference**
- **Chapter 5. Messages for GestIC® Library Update**
- **Appendix A. I²C Command Examples**
- **Appendix B. Glossary**

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENT CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

WARRANTY REGISTRATION

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in the Warranty Registration Card entitles users to receive new product updates. Interim software releases are available at the Microchip website.

RECOMMENDED READING

This user's guide describes how to use MGC3140 GestIC Library Interface. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

MGC3140 3D Tracking and Gesture Controller Data Sheet (40002037)

Consult this document for information regarding the MGC3140 3D Tracking and Gesture Controller.

MXG3141 3D Gesture Controller Data Sheet (40001975)

Consult this document for information regarding the MXG3141 3D Gesture Controller.

Aurea Graphical User Interface User's Guide

Describes how to use the MGC3X30 Aurea Graphical User Interface.

GestIC® Design Guide (DS40001716)

This document describes the GestIC system characteristic parameters and the design process. It enables the user to generate a good electrode design and to parameterize the full GestIC system.

Aurea Software Package – Aurea GUI and GestIC Library

The Aurea GUI contains detailed information on GestIC library features and their parameterization. This information can be accessed via the help pages inside the Aurea parameterization wizard and can also be found as html documents in the Aurea installation folder '01_ Documentation'.

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Information about GestIC technology and MGC3140 can be directly accessed via http://www.microchip.com/gestic.

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB® C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB® REAL ICE™ and MPLAB ICE 2000 in-circuit emulators.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers. This includes MPLAB ICD 3 in-circuit debuggers and PICkit™ 3 debug express.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include production programmers such as MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 in-circuit debugger and MPLAB PM3 device programmers. Also included are nonproduction development programmers such as PICSTART® Plus and PICkit 2 and 3.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers.

Technical support is available through the website at:

http://www.microchip.com/support.

DOCUMENT REVISION HISTORY

Revision C (March, 2021)

Updated Preface; Updated Section 1.1-1.3, 1.6, 2.1, 3.6.3, Section 4.4, and Section 4.5.3.1, 4.5.4.1, 4.5.5.1; Added new Section 4.5.4.2; Removed Section 1.4, 1.5, and 4.5.4.2; Removed UpdateFunction from CRC field description; Updated Table 4-3, 4-7, 4-10, 4-11, and Table A-2 and B-1; Updated Figure 1-1 and 3-1; Updated device name; Updated old terminology; Other minor corrections.

Revision B (April, 2019)

Updated Table 5-2, Table 5-4, Table 5-8, Table 5-11, and Table 5-12.

Other minor corrections.

Revision A (June, 2018)

Initial release of the document.

Chapter 1. Introduction

1.1 PURPOSE OF THIS DOCUMENT

This document is the interface description of the GestIC® Library. It outlines the function of the Library's I²C message interface and contains the complete message reference to control and operate the GestIC system. The interface provides the capability to configure run-time parameters and read back gesture data, positional information and status. More detailed configuration can be performed on design time parameters using the Colibri suite.

The main sections covered are:

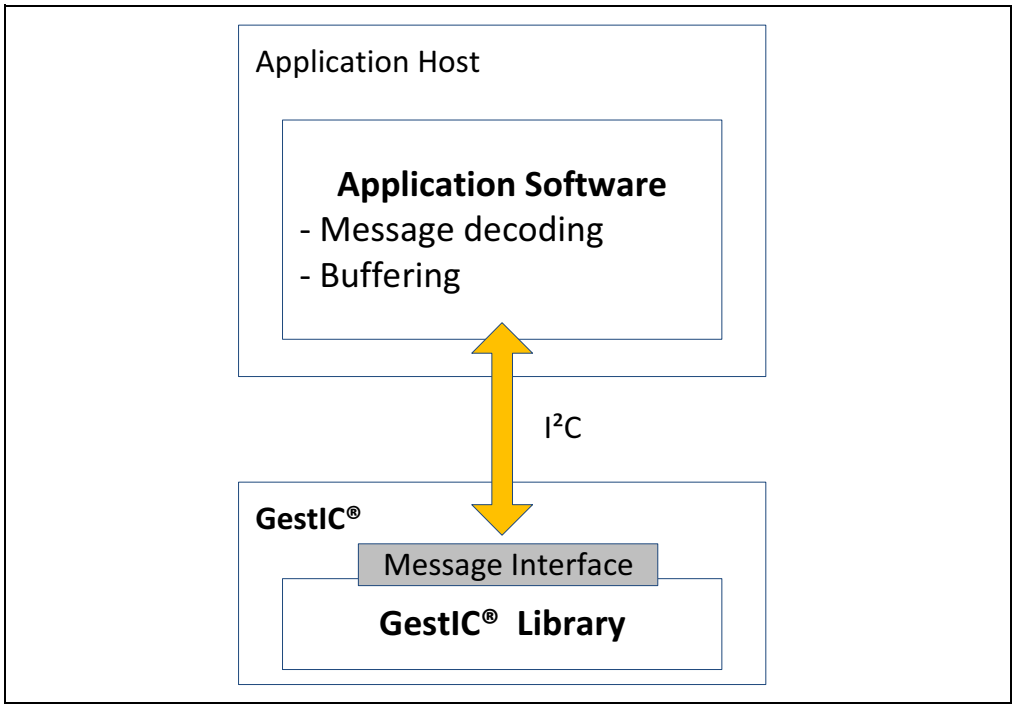
- Description of the message interface and data protocol
- Message reference of the GestIC Library

1.2 GESTIC® SOFTWARE ARCHITECTURE

A GestIC system can be accessed by I²C via the message interface of the GestIC Library.

Figure 1-1 shows the I²C access.

FIGURE 1-1: GESTIC® SOFTWARE ACCESS



Note: The same I^2C interface is used for parameterization and host communication. For parameterization, a connector for the I^2C bridge will be connected. The host must be disconnected or be set to high impedance during parameterization.

1.3 GestIC® LIBRARY

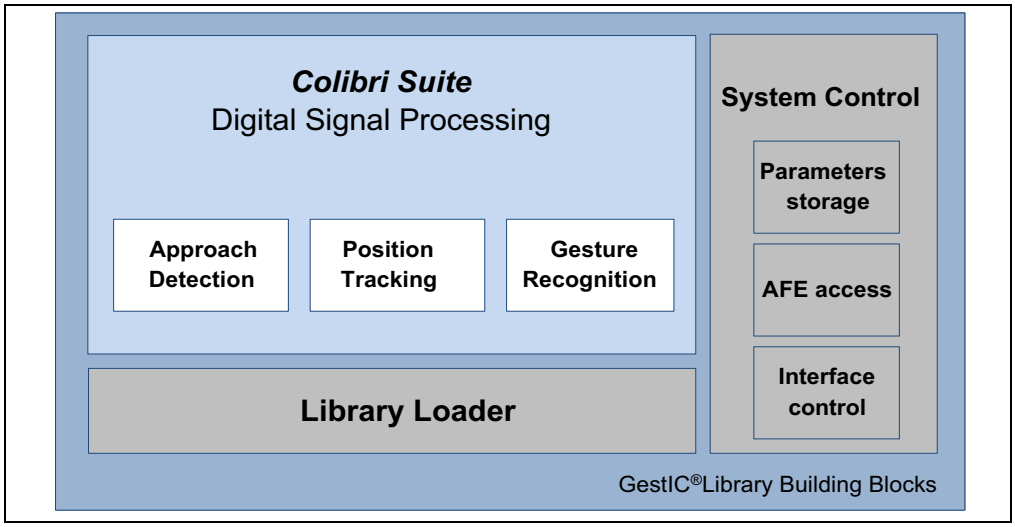
The GestIC Library is embedded firmware stored on the GestIC's internal Flash memory. It contains:

- The Colibri Suite with the digital signal processing algorithms for GestIC features (i.e., GestIC core features Approach Detection, Position Tracking and Gesture Recognition)
- The System Control block handling low-level functions and configuration storage
- The Library Loader for updates of the GestIC Library

The main building blocks are shown in Figure 1-2.

The GestIC Library incorporates a message-based interface that allows the Configuration of the chip and the streaming of the sensor data to the host application.

FIGURE 1-2: BUILDING BLOCKS OF GestIC® LIBRARY



1.4 DEVICE SPECIFIC LIMITATIONS

Note: "Approach Detection", "Airwheel", and "Position Output Information" are supported on the device only if explicitly indicated in the associated hardware data sheet. Enabling these features on devices not supporting them could result in incorrect gesture detection and/or inaccurate position information.

Chapter 2. GestIC® Host Interface

2.1 GESTIC® HARDWARE INTERFACE

Communication with the GestIC is accomplished via a two-wire I^2C-compatible serial port, supported by the TS and $\overline{MCLR}$ lines, so the user can read the sensor data and send control messages to the chip.

Refer to the GestIC data sheet for hardware details on the I^2C interface.

Chapter 3. GestIC® Library Message Interface

3.1 MESSAGES OVERVIEW

GestIC® Library messages are defined for providing sensor data to the host application and for controlling GestIC and its embedded features. They are sent as the payload of the I^2C packets.

TABLE 3-1: MESSAGES FOR SYSTEM CONTROL

ID	Name	Page
0x40	Echo_Request	23
0x15	System_Status	23
0x06	Request_Message	27
0x83	Fw_Version_Info	28
0xA2	Set_Runtime_Parameter	31

TABLE 3-2: MESSAGE FOR SENSOR DATA OUTPUT

ID	Name	Page
0x91	Sensor_Data_Output	40

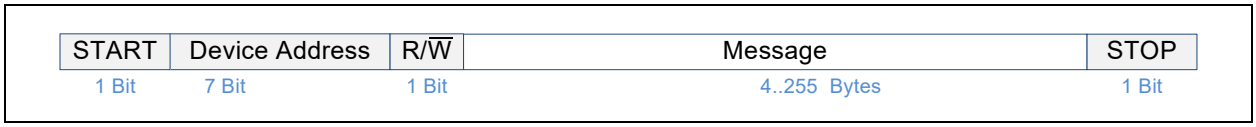
TABLE 3-3: GESTIC® MESSAGES – BOOTLOADER DESCRIPTION

ID	Name	Page
0x70	FwUpdateStart	46
0x71	FwUpdateStartPage	47
0x72	FwUpdateToBuffer	47
0x73	FwUpdateFlashBuffer	48
0x74	FwUpdateVerify	49
0x75	FwUpdateCompleted	50

3.2 MESSAGE FORMAT

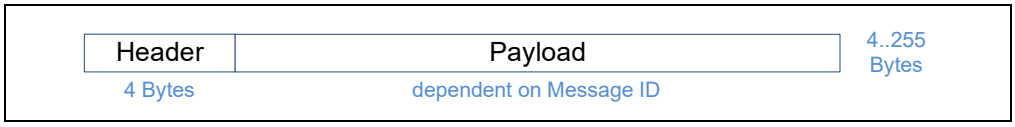
A message is the container to exchange data between the GestIC Library and the application host. Each message has a minimum length of 4 bytes and a maximum of 255 bytes, and fits into the data packets of the communication interface (e.g., I^2C). Each frame transports a single message (see Figure 3-1).

FIGURE 3-1: MESSAGE EMBEDDED IN THE I^2C FRAME



A message consists always of a 4-byte header and a variable payload. The format is shown in Figure 3-2.

FIGURE 3-2: MESSAGE FORMAT



3.3 MESSAGE HEADER

The GestIC Library message header is fixed and has a length of 4 bytes. It contains four data fields, as shown in Figure 3-3 and explained in Table 3-4.

FIGURE 3-3: MESSAGE HEADER

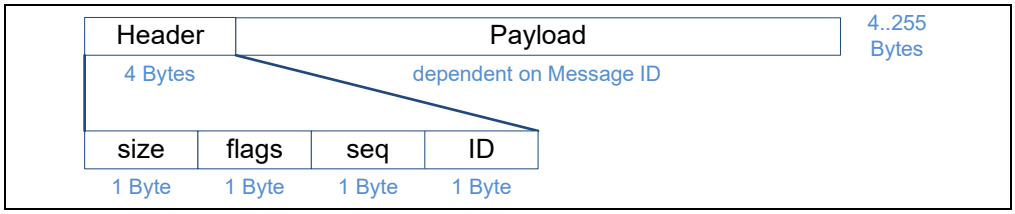


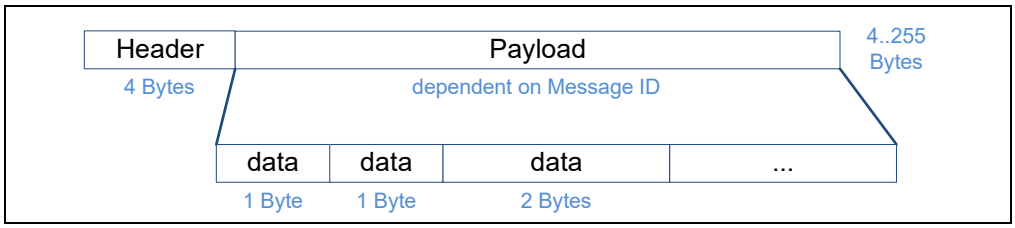
TABLE 3-4: DATA FIELDS OF MESSAGE HEADER

Field	Size (in bytes)	Description
Msg. Size	1	Complete size of the message in bytes including the header.
Flags	1	Reserved for future use.
Seq.	1	Sequence number which is increased for each message sent out by GestIC®. Range is 0…255. The host controller can use that information to verify if the messages got lost during I^2C transmission. GestIC® ignores the sequence number in the received messages.
ID	1	ID of the message. For each ID, the GestIC® Library holds a dedicated structure containing the message direction, its payload elements and possible reply actions.

3.4 MESSAGE PAYLOAD

The message payload has a variable length and consists of one or more payload elements that contain the information to be exchanged. Depending on the content, these elements can be numerical values or dedicated numbers.

FIGURE 3-4: MESSAGE PAYLOAD



Note: Payload elements are exchanged in little-endian format. This means that the Lowest Significant Byte is written first.

Example: Element of 4 bytes: [Byte0]:[Byte1]:[Byte2]:[Byte3]

The structure and content of the payload elements is given in **Chapter 4. "GestIC® Library Message Reference"**.

3.5 MESSAGE CODING AND DECODING

GestIC Library messages can be read as a row of hexadecimal values. In order to decode them, the header and payload elements need to be extracted and mapped to the definition in the message reference (see **Chapter 4. "GestIC® Library Message Reference"**).

As an example message, ID `0x91`, `Sensor_Data_Output` is decoded in the following section.

EXAMPLE 3-1: HEXADECIMAL REPRESENTATION OF MESSAGE `0x91`

```
12 00 15 91 0E 01 EF 80 02 10 00 00 00 00 00 00 00 00
```

3.5.1 Header Extraction

EXAMPLE 3-2: MESSAGE HEADER

```
12 00 15 91 0E 01 EF 80 02 10 00 00 00 00 00 00 00 00
```

The message header contains the following information:

- Size: `0x12` Message including header has a length of 18 bytes
- Flags: `0x00` Flags are not set
- Seq.: `0x15` The message has been sent out with a sequence number of 21
- ID: `0x91` The message ID is `0x91`, `Sensor_Data_Output`

3.5.2 Payload Extraction

EXAMPLE 3-3: MESSAGE PAYLOAD

```
12 00 15 91 0E 01 EF 80 02 10 00 00 00 00 00 00 00 00
```

According to **Section 4.6 "Sensor_Data_Output"**, the payload holds the following data:

- **DataOutputConfigMask** (2 bytes)

  The value of '0x010E' is the bit mask indicating what optional fields are included in the payload. The following bits are set in this example:
  - Bit 1: A GestureInfo field is included in the payload.
  - Bit 2: A TouchInfo field is included in the payload.
  - Bit 3: The AirWheelInfo field is included in the payload.
  - Bit 8 is set but it is one of the reserved bits and must be ignored.

  As all other bits are not set, no DSPStatus, xyzPosition, NoisePower, CICData or SDData fields are included in the message.

- **TimeStamp** (1-byte)

  Value '0xEF' indicates that the event leading to this message occurred when the 200 Hz wrap-around counter was at value 239 (0xEF). This TimeStamp can be used to measure the time difference for events which did not occur too far apart in time (around one second).

- **SystemInfo** (1-byte)

  Value '0x80', which only the DSPRunning flag is set.
  PositionValid is not set so there is either no hand in the electrical field or it is too far away.
  AirWheelValid is not set, so even though there is an AirWheelInfo field included in the message, it should be ignored as currently the AirWheel gesture is not actively detected.
  RawDataValid, NoisePowerValid are also not set, simply because the data is not included in the message.
  The EnvironmentalNoise flag is not set, meaning that there is no external electrical interference which would reduce the performance of the sensor.

- **GestureInfo** (4-bytes)
  This field has value '0x00001002'. So bits <7:0> have the value '0x02' indicating a west to east flick has been performed. And bits <15:12> contain the value '0x1' indicating that the gesture class was *"Flick gesture"*.
  Bit 31 is '0' meaning that the gesture has been completed. If it was '1' it would have meant that the gesture recognition is still in progress.

- **TouchInfo** (4-bytes)
  This field is all zeros in the example, which indicates that the user is not currently touching any electrode.

- **AirWheelInfo** (2-bytes)
  The value is '0x0000' as currently no AirWheel is in progress and the AirWheelValid flag in the SystemInfo field was '0'.

3.6 MESSAGE CONTROL FLOW AND CODING EXAMPLES

3.6.1 Message Control Flow

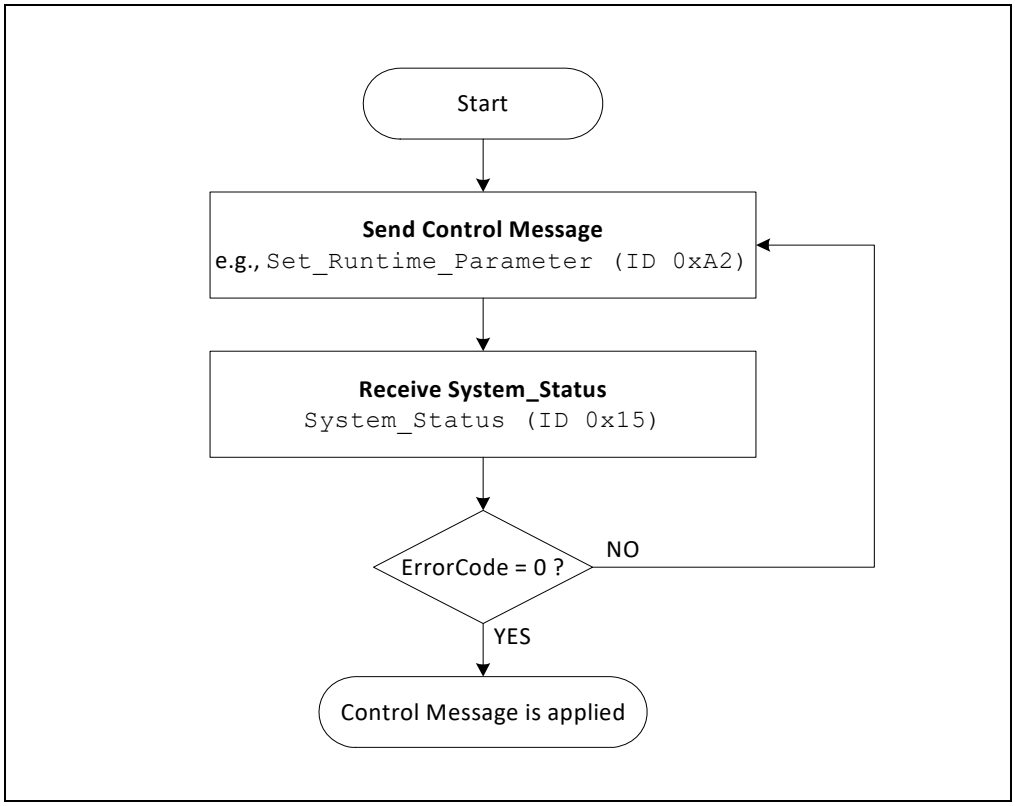
The control of the GestIC Library is done through the following messages:

- `Set_Runtime_Parameter (ID 0xA2)`
- `Request_Message (ID 0x06)`

The GestIC acknowledges each control message by a `System_Status (ID 0x15)`, which contains the original message ID and a 2-byte error code. If the error code is '`0`', the message is applied correctly to GestIC.

The message control flow from the point of view of the application host is shown in Figure 3-5.

FIGURE 3-5: APPLICATION HOST MESSAGE CONTROL



Note: The Emerald I²C to USB bridge prefixes every I²C packet with `0xFE 0xFF` before it is sent out via UART emulation on USB. That is done to allow a frame separation inside the data stream of the PC. For messages sent to GestIC® via the I²C to USB bridge from a terminal program (e.g., Hterm), the prefix has to be added as well.

3.6.2 Read GestIC® Library Version

After Power-on or Reset, GestIC runs the Library Loader and sends out the message `Fw_Version_Info (0x83)`. The application host can receive this message as a first communication check. After a time-out of 200 ms, the GestIC Library Processing mode is started automatically.

The application host can request the `FW_Version_Info` during run time by using `Request_Message (0x06)`.

3.6.2.1 EXAMPLE: `Request_Fw_Version INFO`

The following example shows how the `Request_Message (0x06)` is used to request a `FW_Version_Info (0x83)` message.

TABLE 3-5: MESSAGE FROM HOST TO MGC3140: `REQUEST_MESSAGE (0X06)`

Raw Message	0C 00 00 06 83 00 00 00 00 00 00 00		
Payload Element	MessageID	Reserved	Parameter
Hex in little-endian	83	00 00 00	00 00 00 00
Hex decoded	0x83	n/a	n/a
Description	FW_Version_Info	n/a	n/a

MGC3140 replies with message `FW_Version_Info (0x83)` followed by `System_Status (0x15)`, containing the error code.

TABLE 3-6: MESSAGE FROM MGC3140 TO HOST: `FwVersionInfo (0X83)`

Raw Message	84 00 01 83 XX

3.6.3 Run-Time Control

A dedicated set of run-time control options is provided within the message `Set_Runtime_Parameter (0xA2)`. It can be used to control the active feature set and sensor data output and, thus, it allows the build-up of a context-sensitive operation of GestIC. For a detailed message description, refer to **Section 4.5 "Set_Runtime_Parameter"**.

The following examples show how to set relevant run-time parameters.

3.6.3.1 EXAMPLE: ENABLE APPROACH DETECTION

This example shows how to enable the Approach Detection mode by using the message `Set_Runtime_Parameter (0xA2)`. (Only on supported devices.)

TABLE 3-7: MESSAGE FROM HOST TO MGC3140: SET_RUNTIME_PARAMETER (0XA2)

Raw Message	10 00 00 A2	97 00	00 00	01 00 00 00	01 00 00 00
Payload Element	RuntimeParameterID	Reserved	Argument0	Argument1
Hex in little-endian	97 00	00 00	01 00 00 00	01 00 00 00
Hex decoded	0x0097	n/a	0x00000001	0x00000001
Description	ApproachDetection	n/a	Enable Approach Detection mode	Mask for Approach Detection bit

MGC3140 replies with message `System_Status (0x15)`, containing the error code.

Note: The Approach Detection mode is not available for MXG3141.

TABLE 3-8: MESSAGE FROM MGC3140 TO HOST: SYSTEM_STATUS (0X15)

Raw Message	10 00 08 15	A2	34	00 00	00 00 00 00	00 00 00 00
Payload Element	MsgID	MaxCmdSize	ErrorCode	Reserved	Reserved
Hex in little-endian	A2	34	00 00	00 00 00 00	00 00 00 00
Hex decoded	0xA2	0x34	0x0000	n/a	n/a
Description	Acknowledge to ID 0xA2	n/a	No error	n/a	n/a

3.6.3.2 EXAMPLE: ENABLE ALL GESTURES

This example shows how to enable all gestures (Flicks and Circles) by using the message `Set_Runtime_Parameter (0xA2)`.

TABLE 3-9: MESSAGE FROM HOST TO MGC3140: SET_RUNTIME_PARAMETER (0XA2)

Raw Message	10 00 00 A2	85 00	00 00	7F 00 00 00	7F 00 00 00
Payload Element	RuntimeParameterID	Reserved	Argument0	Argument1
Hex in little-endian	85 00	00 00	7F 00 00 00	7F 00 00 00
Hex decoded	0x0085	n/a	0x0000007F	0x0000007F
Description	despGestureMask	n/a	Enable gestures 0...6	Mask for Enable gestures 0...6 bits

MGC3140 replies with message `System_Status (0x15)`. Refer to Table 3-8.

3.6.3.3 EXAMPLE: ENABLE DATA OUTPUT

This example shows how to enable the sensor data output of Gesture Data, Touch Data, AirWheel Data and Position Data. Refer to **Section 4.5.5.4 "Data Output Enable Mask"**.

TABLE 3-10: MESSAGE FROM HOST TO MGC3140: SET_RUNTIME_PARAMETER (0XA2)

Raw Message	10 00 00 A2	A0 00	00 00	1E 00 00 00	FF FF FF FF
Payload Element	RuntimeParameterID	Reserved	Argument0	Argument1	
Hex in little-endian	A0 00	00 00	1E 00 00 00	FF FF FF FF	
Hex decoded	0x00A0	0x0000	0x0000001E	0xFFFFFFFF	
Description	DataOutputEnableMask	n/a	Enable bit 1...bit 4; disable all other bits	Overwrite existing Configuration	

MGC3140 replies with message System_Status (0x15). Refer to Table 3-8.

3.6.3.4 EXAMPLE: LOCK DATA OUTPUT

This example shows how to lock the sensor data output of Gesture Data, Touch Data, AirWheel Data and Position Data. Refer to **Section 4.5.5.5 "Data Output Lock Mask"**.

TABLE 3-11: MESSAGE FROM HOST TO MGC3140: SET_RUNTIME_PARAMETER (0XA2)

Raw Message	10 00 00 A2	A1 00	00 00	1E 00 00 00	FF FF FF FF
Payload Element	RuntimeParameterID	Reserved	Argument0	Argument1	
Hex in little-endian	A1 00	00 00	1E 00 00 00	FF FF FF FF	
Hex decoded	0x00A1	0x0000	0x0000001E	0xFFFFFFFF	
Description	DataOutputLockMask	n/a	Enable bit 1...bit 4; disable all other bits	Overwrite existing Configuration	

MGC3140 replies with message System_Status (0x15). Refer to Table 3-8.

3.6.4 Sensor Data Output

The GestIC Library processes sensor data with a default update rate of 5 ms. That means the I²C message buffer is regularly updated in that time interval. Whenever new data is available, GestIC pulls the TS line to request the I²C host to transfer this data. Sensor data sent from GestIC to the host are included in the message `Sensor_Data_Output (0x91)`.

The content of the sensor data output can be configured via the message `Set_Runtime_Parameter (0xA2)`.

3.6.4.1 EXAMPLE: READ SENSOR DATA OUTPUT

In the following examples the sensor data output is configured according to **Section 3.6.3.3 "Example: Enable Data Output"** and **Section 3.6.3.4 "Example: Lock Data Output"**.

TABLE 3-12: MESSAGE FROM MGC3140 TO HOST: FLICK EAST TO WEST

Raw Message	18 08 FF 91 1E 01 57 8C 03 10 04 00 00 00 00 00 00 00 00 00 00 00 00

Payload Element	SystemInfo	GestureInfo	TouchInfo	Air-WheelInfo	xyzPosition
Hex in little-endian	8C	03 10 04 00	00 00 00 00	00 00	00 00 00 00 00 00
Hex decoded	0x8C	0x00041003	0x00000000	0x0000	0x000000000000
Description	Bit 2: RawDataValid Bit 3: NoisePowerValid Bit 7: DSPRunning	Flick East to West	No touch	No AirWheel	No Position Data available

TABLE 3-13: MESSAGE FROM MGC3140 TO HOST: TOUCH OF CENTER ELECTRODE

Raw Message	18 08 3B 91 1E 01 38 8D 00 00 00 00 10 00 00 00 00 00 5A A6 12 53 6B 0A

Payload Element	SystemInfo	GestureInfo	TouchInfo	Air-WheelInfo	xyzPosition
Hex in little-endian	8D	00 00 00 00	10 00 00 00	00 00	5A A6 12 53 6B 0A
Hex decoded	0x8D	0x00000000	0x00000010	0x0000	Byte 1 and 2: 0xA65A Byte 3 and 4: 0x5312 Byte 5 and 6: 0x0A6B
Description	Bit 0: PositionValid Bit 2: RawDataValid Bit 3: NoisePowerValid Bit 7: DSPRunning	No Gesture Detected	Touch on Center Electrode	No AirWheel Data	x: 42586 y: 21266 z: 2667

TABLE 3-14: MESSAGE FROM MGC3140 TO HOST: POSITION

Raw Message	18 08 44 91 1E 01 41 8D 00 00 00 00 00 00 00 00 00 00 2F B2 E7 87 6A 35				
Payload Element	SystemInfo	GestureInfo	TouchInfo	Air-WheelInfo	xyzPosition
Hex in little-endian	8D	00 00 00 00	00 00 00 00	00 00	2F B2 E7 87 6A 35
Hex decoded	0x8D	0x00000000	0x00000000	0x0000	Byte 1 and 2: 0xB22F Byte 3 and 4: 0x87E7 Byte 5 and 6: 0x356a
Description	Bit 0: PositionValid Bit 2: RawDataValid Bit 3: NoisePowerValid Bit 7: DSPRunning	No Gesture Detected	Touch on Center Electrode	No AirWheel Data	x: 45615 y: 34791 z: 13674

Chapter 4. GestIC® Library Message Reference

4.1 ECHO_REQUEST

When the Echo_Request message is sent to the chip, GestIC will reply with the same content. Message length and content can be chosen by the user. For a correct communication, the total length of the sent message must be entered in the size field of the message header. The sequence looks like the following:

- Sent: 0A 00 00 40 01 02 03 04 05 06
- Reply: 0A 00 68 40 01 02 03 04 05 06

Direction: Host to GestIC, reply GestIC to Host

TABLE 4-1: ECHO_REQUEST MESSAGE OVERVIEW

Header				Payload
Msg. Size	Flags	Seq.	ID	User-defined content
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*n/a*
n/a	*n/a*	*n/a*	0x40	Zero or more octets of arbitrary data

4.2 SYSTEM_STATUS

System_Status is used to acknowledge the reception of messages from the host. This message holds the error code and is used to confirm the transmission of the following messages:

- Request_Message
- Set_Runtime_Parameter
- FwUpdateStart
- FwUpdateStartPage
- FwUpdateToBuffer
- FwUpdateFlashBuffer
- FwUpdateVerify
- FwUpdateCompleted

The message format has been changed compared to MGC3x30 variants.

Direction: GestIC to Host

TABLE 4-2: MESSAGE OVERVIEW

Header				Payload						
Msg. Size	Flags	Seq	ID	MsgId	MaxCmdSize	ErrorCode	Reserved1	Flags	SeqCtr	Reserved2
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*	*2 Bytes*	*1 Byte*	*1 Byte*	*4 Bytes*
0x10	*n/a*	*n/a*	0x15	**see description below**						

TABLE 4-3: PAYLOAD ELEMENTS

Element	Size	Description
MsgId	1	If the status message is in response to a command from the host, then the MsgId (including message header) is accepted by the firmware
MaxCmdSize	1	Maximum allowed message size which the firmware will accept
ErrorCode	2	Error code, referring to the last received message ***Structure***: 16-bit Word containing dedicated values (see list below) Possible Values: These error codes are sent by the Library Loader, Library Loader Updater and Library: 0x0000 NoError — OK 0x0001 UnknownCommand — The MsgId is unknown These error codes are sent by the Library Loader: 0x0002 InvalidSessionId — Session ID is invalid or does not match (0x0 is not allowed) (message FwUpdateStart, FwUpdateCompleted) 0x0003 InvalidMsgCrc — CRC of the current message is invalid, (message FwUpdateBlock, FwUpdateStart, FwUpdateCompleted) 0x0004 InvalidLength — Length is invalid (message FwUpdateBlock) 0x0005 InvalidAddress — Address is invalid (message FwUpdatedBlock) 0x0006 InvalidFunction — Function ID is invalid (message FwUpdateStart, FwUpdatedBlock) 0x0008 ContentMismatch — VerifyOnly function found a mismatch between content and Flash memory (message FwUpdateBlock) 0x0009 NoClientReachable — A client is not available, or communication is lost 0x000A NoFwPresent — No valid Fw is present to execute

TABLE 4-3: PAYLOAD ELEMENTS (CONTINUED)

Element	Size	Description
ErrorCode	2	0x000B WrongParameterAddr — Parameter address does not match Bootloader assumption 0x000C WrongChip — Parameter address does not match Bootloader assumption 0x000D InvalidBufferCrc — CRC of the page buffer is invalid, (message FwUpdateFlashBuffer) 0x000E DataTooLong — Data too long, FwUpdateToBuffer exceeds buffer size 0x000F SessionInitFailed — Failed to initialize session (message FwUpdateStart) 0x0010 VerifyOK — Verify OK (message: FwUpdateVerify) 0x0011 UnpermittedOperation — Unpermitted operation over the current page MsgIdFwUpdateFlashBuffer on FwInfo Page These error codes are sent by the Library: 0x0014 WrongParameterValue — Value of the Argument/Parameter of a RuntimeParameter command is out of the valid range (message: Request_Message and Set_Runtime_Parameter) 0x0015 UnknownParameterID — MsgId or RuntimeParameterId is unknown or out of the valid range (message: Request_Message and Set_Runtime_Parameter) 0x0016 CompareAfterProgrammingFailed — Compare After Flash Programming Failed 0x001A WakeupHappened — A wake-up by Host was detected These error codes are sent by the Library Loader Updater: 0x0080 LoaderUpdateStarted — Bootloader update started 0x0081 LoaderUpdateFinished — Bootloader update finished 0x0082 LoaderUpdateFailed — Bootloader update failed 0x008E WrongChipID — Chip ID wrong for update 0x008F CommandTooShort — Command too short 0x0090 BadChecksum — I^2C checksum error 0x0091 BadAppChecksum — App checksum error 0x0092 FlashPageNotEmptyAfterErase — Flash page not empty after erase 0x0093 FlashPageMismatchAfterWrite — Flash page mismatch after write 0x0094 FlashEraseRangesNotSupported — Flash erase ranges not supported

TABLE 4-3: PAYLOAD ELEMENTS (CONTINUED)

Element	Size	Description
Reserved1	2	Reserved for future use
Flags	1	Copy of the flags field from the header of the last received message
SeqCtr	1	Copy of the sequence counter field from the header of the last received message
Reserved2	4	Reserved for future use

4.3 REQUEST_MESSAGE

`Request_Message` forces GestIC Library to reply to the message with the requested ID.

Direction: Host to GestIC

TABLE 4-4: MESSAGE OVERVIEW

Header				Payload		
Msg. Size	Flags	Seq.	ID	MessageID	Reserved	Param.
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*3 Bytes*	*4 Bytes*
0x0C	*n/a*	*n/a*	0x06	Refer to Table 4-5		

TABLE 4-5: PAYLOAD ELEMENTS

Element	Size (bytes)	Description
MessageID	1	Request the Message with ID, MessageID, from GestIC® Library. GestIC® Library will answer with the requested message or stay silent. ***Structure***: Single-byte read as a hexadecimal value ***Range:*** (0x00...0xFF)
Reserved	3	Reserved, write as '0'.
Param.	4	Optional, parameter can be used to specify the kind of return. Example: Requesting message SetRuntimeParameter, Param. specifies the RuntimeParameterId to read back the parameter. ***Structure***: 32-bit Word, containing dedicated values or bit fields. ***Range:*** (0x00000000...0xFFFFFFFF)

Note 1: The `Request_Message` command can only be used with `MessageID` 0x83 and 0xA2.

2: The `TransFreqSelect` run-time parameter is a write-only parameter and cannot be requested with message `Request_Message`.

3: For examples of the `Request_Message` command, refer to Table A-1.

4.4 FW_VERSION_INFO

The FirmwareVersion message contains detailed version information for GestIC bootloader, firmware and parameterization. The message format has been changed compared to MGC3x30 variants. MGC3140/MXG3141 will transmit this message after a Reset.

Direction: MGC3140/MXG3141 to Host.

TABLE 4-6: MESSAGE OVERVIEW

Header				Payload										
Msg. Size	Flags	Seq.	ID	FwValid	HwRev	ParameterPage	BootloaderMinor	BootloaderMajor	ChipId	FirmwareStartPage	VersionString	CustomString	NewStructIndicator	FwInfoMajor
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*9 Bytes*	*16 Bytes*	*3 Bytes*	*1 Byte*
0x84	*n/a*	*n/a*	0x83	**Refer to Table 4-7**										

Payload														
FwInfoMinor	FwMajor	FwMinor	FwRev	Padding0	CommitDistance	RcFwType	RcFwFlags	RcFwGitHash	RcDspType	RcDspFlags	RcDspPad0	RcDspRevision	RcDspPad1	BiEpoch
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*	*1 Byte*	*1 Byte*	*14 Bytes*	*1 Byte*	*1 Bytes*	*2 Bytes*	*4 Bytes*	*8 Bytes*	*4 Byte*
Refer to Table 4-7														

Payload									
BiFlags	BiUserId	BiPadding	SysClkHz	IdDspId	IdParameterId	IdApplicationId	IdAppDetail	PadToBootInfo	Reserved
1 Byte	*1 Byte*	*2 Bytes*	*4 Bytes*	*2 Bytes*	*2 Bytes*	*2 Bytes*	*2 Bytes*	*12 Bytes*	*20 Bytes*
Refer to Table 4-7									

TABLE 4-7: PAYLOAD ELEMENTS

Element	Size (bytes)	Description
FwValid	1	This field indicates if the firmware in Flash is valid or somehow corrupted. All values different from the defined should be treated as invalid. ***Possible values***: 0x00 Empty0 No complete firmware image on the device 0x0A InvalidFw Firmware on-chip is invalid. Indicates a started update that has been interrupted 0xAA ValidFw A valid complete image is stored in the device Flash 0xFF Empty1 No complete firmware image on device
HwRev	2	Hardware revision
ParameterPage	1	First Flash page of the parameter section
BootloaderMinor	1	Bootloader version, minor digit(1)
BootloaderMajor	1	Bootloader version, major digit(1)
ChipId	1	Identifies the device on which this firmware runs ***Possible values***: 0x14 Sabrewing Device is a 3130 (Sabrewing V1.0 variant) 0x15 Hillstar Device is a 3130 (Hillstar variant) 0x32 Woodstar Device is a 3030 (Woodstar variant) 0x41 3140 Device is a 3140 (Emerald variant) 0x42 3141 Device is a 3141 (2D/3D)
FirmwareStartPage	1	Page containing the firmware entry point.
VersionString	9	Version in format 'major.minor.rev' as string, filled with ';' until the 9 characters are full.
CustomString	16	16 characters for customer usage. If the string is shorter than 16 characters it must be filled with spaces. Entries defined by Microchip start with 'MCHP:'
NewStructIndicator	3	Contains the three character sequence {';', '!', ';\0'} which is used to indicate that the firmware version info conforms to the format described here.
FwInfoMajor	1	Major version of this FirmwareVersion message structure. To confirm with this description it must be '1'.
FwInfoMinor	1	Minor version of this FirmwareVersion message structure. To confirm with this description it must be '0'.
FwMajor	1	8-bit integer containing the 'major' field of firmware version (firmware version is in format 'major.minor.rev').
FwMinor	1	8-bit integer containing the 'minor' field of firmware version (firmware version is in format 'major.minor.rev').
FwRev	1	8-bit integer containing the 'Rev' field of firmware version (firmware version is in format 'major.minor.rev').
Padding0	1	Reserved field, used to align the following field properly.
CommitDistance	2	Internal version information, describes how many commits we are away from the last set version tag in the VCS.
RcFwType	1	Revision control type of firmware, always '2' as GestIC® sources use 'Git'.
RcFwFlags	1	Revision control info for firmware: 'Flags' field. Bit 0 indicates that the sources where modified when compiling; Bit 1 indicates that the compile was from multiple mixed revisions.
RcFwGitHash	14	Revision control info for firmware: First 14 bytes of git hash for firmware.
RcDspType	1	Revision control type of DSP, always '1' as DSP sources use SVN.
RcDspFlags	1	Revision control info for DSP: 'Flags' field. Bit 0 indicates that the sources where modified when compiling; Bit 1 indicates that the compile was from multiple mixed revisions.

TABLE 4-7: PAYLOAD ELEMENTS (CONTINUED)

Element	Size (bytes)	Description
RcDspPad0	2	Revision control info for DSP: Padding to align next field containing the SVN revision.
RcDspRevision	4	Revision control info for DSP: Revision number from SVN.
RcDspPad1	8	Revision control info for DSP: Another 8 bytes of padding to fill generic revision info structure.
BiEpoch	4	BuildInfo: Build time UTC in 'epoch' format.
BiFlags	1	BuildInfo: Flags, if Bit 0 is set the user who builds the file is known and the 'UserId BuildInfo' field contains a valid value.
BiUserId	1	BuildInfo: Numeric ID identifying the user who build this firmware.
BiPadding	2	BuildInfo: Pad build info length to 8 bytes. Reserved for future use.
SysClkHz	4	System clock in Hz.
IdDspId	2	ID: DSP id field (900x = MGC3130, 910x = MGC3030, 45 = Sabrewing, 440x = MGC3140, 45xx = MXG3141).
IdParameterId	2	ID: Parameter Id can be used to identify different parameterizations.
IdApplicationId	2	ID: Application Id can be used to differentiate different applications. Currently only '0' (regular FW) and '1' (bootloader updater) are defined.
IdAppDetail	2	ID: 'appDetail' is currently a reserved field for future use. Must be '0'.
PadToBootInfo	12	Currently reserved bytes, used to pad until start of 'BootInfo' information.
Reserved	20	Reserved for future use.

Note 1: The current bootloader version is only contained when the message is "self-emitted" after Reset. When the message is requested by Request_Message, the bootloader version is always reported 0.0.

4.5 SET_RUNTIME_PARAMETER

This message is used to set run-time parameters within the GestIC Library. It supports parameters for AFE parameterization, feature Configuration, and sensor data output. A special value is defined for a persistent saving of parameters to the Flash memory. Parameters which can be made persistent are grouped into three categories:

- **Analog Front-End (AFE) Category**
- **Digital Signal Processing (DSP) Category**
- **System Category**

Direction: Host to GestIC

TABLE 4-8: MESSAGE OVERVIEW

Header				Payload			
Msg. Size	Flags	Seq.	ID	RuntimeParameterID	Reserved	Argument0	Argument1
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*	*2 Bytes*	*4 Bytes*	*4 Bytes*
0x10	*n/a*	*n/a*	0xA2	see description below			

TABLE 4-9: PAYLOAD ELEMENTS

Element	Element Size (bytes)	Description
RuntimeParameterID	2	ID of run-time parameter. Refer to **Section 4.5.1 "Trigger"** through **Section 4.5.5.5 "Data Output Lock Mask"**. ***Structure*:** 16-bit Word interpreted as hex value ***Range:*** (0x0000...0xFFFF)
Reserved	2	Write as '0'
Argument0	4	Argument values, depending on run-time parameter ID. If not used, Argument0 should be provided as '0'. ***Structure***: 32-bit Word: Argument0 ***Range***: depends on run-time parameter
Argument1	4	Argument values, depending on run-time parameter ID. If not used, Argument1 should be provided as '0'. ***Structure***: 32-bit Word: Argument1 ***Range***: depends on run-time parameter

4.5.1 Trigger

This parameter forces a trigger defined in `Argument0`.

`RuntimeParameterID`	`0x1000`	`Trigger:` Parameter forces a trigger
`Argument0`	`0x00000000`:	Force re-calibration
	`0x00000002`:	Enter Deep Sleep 1: The wake-up sources from Deep Sleep 1 are I^2C0 Start bit detection or MCLR Reset. The system will resume from Deep Sleep on any I^2C messages sent on the bus, and the first I^2C message will be lost.
	Range: (`0x00000000`, `0x00000002`)	
`Argument1`	Not used	

4.5.2 Make Persistent

Use this ID to make the parameter set defined in `Argument0` persistent (store to Flash memory).

`RuntimeParameterID`	`0xFF00`	`MakePersistent:` Stores parameter in Flash
`Argument0`	`0x00000000`:	Store RTPs for AFE Category
	`0x00000001`:	Store RTPs for DSP Category
	`0x00000002`:	Store RTPs for System Category
	Range: (`0x00000000`, `0x00000001`, `00000002`)	
`Argument1`	Not used	

4.5.3 Analog Front-End (AFE) Category

4.5.3.1 ELECTRODE MAPPING

Electrode mapping indicates what physical channel is used for a given logical electrode.

There are five "RuntimeParameterID" that can be associated to any of the five physical receive channels (device "RXn" pins).

Depending on how the device has been configured, electrodes can be intended in Frame mode or 4 South + 1 North mode.

Refer to the device hardware data sheet for a description of the available styles and for the location of the RX pins on the chosen package.

TABLE 4-10: ELECTRODES MAPPING TO RUNTIMEPARAMETERID

RuntimeParameterID	Electrode Name	
Value	**Frame Style**	**4S + N Style**
0x65	South	S1
0x66	West	S2
0x67	North	S3
0x68	East	S4
0x69	Center	N
Argument0	Contains the number of physical receive channels (Rx0, Rx1, Rx2, Rx3, Rx4) ***Range:*** (0x00000000, 0x00000001, 0x00000002, 0x00000003, 0x00000004)	
Argument1	Not used	

4.5.4 Digital Signal Processing (DSP) Category

4.5.4.1 TRANSMIT FREQUENCY SELECTION

This sets the total number of transmitter frequencies used, and the order in which they are tested for the frequency hopping.

`RuntimeParameterID`	`0x82`	`TransFreqSelect`: Parameter to set the frequency IDs used
`Argument0`		Total number of used Tx frequencies. This parameter can be 1, 2, 3, 4 or 5.
`Argument1`		This determines in what order the transmitter frequencies are tested. The indexes are numbered 0 to 4 and represent respective transmitter frequencies. For MXG3141, the default frequencies are 44 kHz and 64/65/66/67 kHz.
Example:		e.g., `Argument0 = 0x04` in combination with `Argument1 = 0x3104` means that frequencies with the index 4, 0, 1 and 3 are used and tested in this specific order. e.g., Index – Default Frequency Mapping (`Argument 0 = 0x5,Argument 1 = 0x43210`)

Note: The `TransFreqSelect` run-time parameter is a write-only parameter and could not be requested with `REQUEST_MESSAGE (0x06)` message.

The active frequency is reported in the `SensorDataOutput` element '`DSPStatus`' whenever the frequency changes.

4.5.4.2 TOUCH DETECTION

This parameter enables/disables Touch Detection.

`RuntimeParameterID`	`0x97`	`dspTouchConfig`: Parameter to enable/disable Touch Detection
`Argument0`		Set `Argument0` to '`0x08`' to enable Touch Detection Set `Argument0` to '`0x00`' to disable Touch Detection **Note:** If `Argument1` is not set correctly, the system will show malfunctions.
`Argument1`	`0x08`	

4.5.4.3 APPROACH DETECTION

This parameter enables/disables Approach Detection mode. (Only on supported devices.)

`RuntimeParameterID`	`0x97`	`dspApproachDetectionMode`: Parameter to enable/ disable Approach Detection Mode
`Argument0`		Set `Argument0` to `0x01` to enable Approach Detection Set `Argument0` to `0x00` to disable Approach Detection **Note:** If `Argument1` is not set correctly, the system will show malfunctions.
`Argument1`	`0x01`	

Note: The Approach Detection mode is not available for MXG3141.

4.5.5 System Category

4.5.5.1 AIRWHEEL

This parameter enables/disables AirWheel. (Only on supported devices.)

RuntimeParameterID	0x90	dspAirWheelConfig: Parameter to enable/disable AirWheel
Argument0		Set Argument0 to '0x20' to enable AirWheel Set Argument0 to '0x00' to disable AirWheel **Note:** If Argument1 is not set correctly, the system will show malfunctions.
Argument1	0x20	

Note: AirWheel is not supported by MXG3141.

4.5.5.2 GESTURE PROCESSING (HMM)

This parameter enables the in-built gestures. Disabling one gesture will increase the recognition probability of the others.

If a bit in Argument0 is set to '1', the respective Gesture will be enabled. If a bit in Argument0 is set to '0', the respective Gesture will be disabled.

RuntimeParameterID	0x85	dspGestureMask: Parameter to enable/disable gestures
Argument0		Bit 0: Garbage model Bit 1: Flick West to East Bit 2: Flick East to West Bit 3: Flick South to North Bit 4: Flick North to South Bit 5: Circle clockwise Bit 6: Circle counterclockwise Bit 7: Reserved Bit 8: Reserved Bit 22: Hold gesture Bit 23: Presence gesture Bit 24: Edge Flick West to East Bit 25: Edge Flick East to West Bit 26: Edge Flick South to North Bit 27: Edge Flick North to South Bit 28: Double Flick West to East Bit 29: Double Flick East to West Bit 30: Double Flick South to North Bit 31: Double Flick North to South
Argument1		Acts as a mask, set appropriate bits to '1' to change the flag. All other flags remain unchanged.

Note: Circle and Double Flick are not supported by MXG3141.

4.5.5.3 CALIBRATION OPERATION MODE

This parameter enables/disables the selected auto-calibration feature.

If a bit in `Argument0` is set to '`0`', the respective auto-calibration feature will be enabled.

If a bit in `Argument0` is set to '`1`' the respective auto-calibration feature will be disabled.

`RuntimeParameterID`	`0x80`	`dspCalOpMode`: Parameter to enable/disable auto-calibration
`Argument0`		Bit 1: Enable/disable gesture-triggered calibration
		Bit 2: Enable/disable negative calibration
		Bit 3: Enable/disable idle calibration
		Bit 4: Enable/disable invalidity value calibration, if values are completely out of range
		Bit 5: Enable/disable calibration triggered by AFA
`Argument1`		Acts as a mask, set appropriate bits to '`1`' to change the flag. All other flags remain unchanged.

4.5.5.4 DATA OUTPUT ENABLE MASK

This parameter determines the data output of the message `Sensor_Data_Output (0x91)`. If a bit in `Argument0` is set to '`1`', the respective payload element will be part of the message `Sensor_Data_Output (0x91)`. If a bit in `Argument0` is set to '`0`', the payload element will not be part of the message `Sensor_Data_Output (0x91)` when the data is updated (payload element is 'Off').

Use `DataOutputEnableMask` to optimize the sensor data output in terms of I^2C utilization and efficiency of the host code.

Note: Enabling all payload elements might lead to malfunctions due to bandwidth limitations on the I^2C bus.

`RuntimeParameterID` `0xA0` `DataOutputEnableMask`: Parameter determining the data output

`Argument0`

Bits 0...12: Payload elements: If set to '`1`', payload elements will be part of the message

- Bit 0: DSP Status
- Bit 1: Gesture Data
- Bit 2: TouchInfo
- Bit 3: AirWheelInfo
- Bit 4: xyzPosition
- Bit 5: Noise Power
- Bit 6...10: These bits are reserved and must be set to '`0`'
- Bit 11: CICData (Uncalibrated Signal)
- Bit 12: SDData (Signal Deviation)

Bits 13...15: These bits are reserved and must be set to '`0`'

Bits 16...17: SystemInfo Status bits: If set to '`1`', the reporting of a state change in the payload element SystemInfo is enabled

- Bit 16: EnvironmentalNoise indication
- Bit 17: Clipping indication**(1)**

Bit 18: DSP running

Bits 19: AirWheelCounterDecimation: If set to '`1`', the AirWheel counter is decimated by the factor of 4

Bit 20: TimeStampOverflow:
This applies when AirWheel or Touch Detection is ongoing.
If activated, a message will be sent when the counter in the payload element TimeStamp is overflowing (TimeStamp=`0`)

Bits 21...26: These bits are reserved

Bits 27...31: GesturesInfo Status bits: If set to '`1`', the reporting of a state change in the payload element GestureInfo is enabled.

- Bit 27: HandPresence flag
- Bit 28: HandHold flag
- Bit 29: HandInside flag

Bit 30: This bit is reserved

Bit 31: GestureInProgress flag

`Argument1` Acts as a mask, set appropriate bits to '`1`' to change the flag. All other flags remain unchanged.

Note 1: Clipping indication is not supported by MGC3140/MXG3141. Message is kept for compliance with MGC3x30

4.5.5.5 DATA OUTPUT LOCK MASK

This parameter determines the data output of the `Sensor_Data_Output (0x91)` message. If a bit in `Argument0` is set to '`1`', the respective payload element will be part of the `Sensor_Data_Output (0x91)` message, no matter whether there is new data or not (payload element is 'On').

If a bit in `Argument0` is set to '`0`', the payload element will only be part of the message `Sensor_Data_Output (0x91)` when the data is updated (payload element is 'Dynamic').

`RuntimeParameterID`	`0xA1`	`DataOutputLockMask`: Parameter determining the data output
`Argument0`		Bits 0...12: Payload elements: If set to '`1`', payload elements will be part of the message
		Bit 0: DSP Status
		Bit 1: Gesture Data
		Bit 2: TouchInfo
		Bit 3: AirWheelInfo
		Bit 4: xyzPosition
		Bit 5: Noise Power
		Bit 6...10: These bits are reserved and must be set to '`0`'.
		Bit 11: CICData (Uncalibrated Signal)
		Bit 12: SDData (Signal Deviation)
		Bits 13...15: These bits are reserved and must be set to '`0`'
`Argument1`		Acts as a mask, set appropriate bits to '`1`' to change the flag. All other flags remain unchanged.

4.5.5.6 DATA OUTPUT REQUEST MASK

This parameter determines the data output only of the next message `Sensor_Data_Output (0x91)`. If a bit in `Argument0` is set to '`1`', the respective payload element will be part of the next message `Sensor_Data_Output (0x91)`.

If a bit in `Argument0` is set to '`0`', the payload element will not be part of the next message `Sensor_Data_Output (0x91)` when the data is updated.

This will force the GestIC to send a new message `Sensor_Data_Output (0x91)` even if there were no valid events and data. This message will contain data according to the `Argument0` selection. Then the `Sensor_Data_Output (0x91)` will be sent according to the Data Output Enable and Lock masks only on valid events and data.

`RuntimeParameterID`	`0xA2`	`DataOutputRequestMask`: Parameter determining the next data output
`Argument0`		Bits 0...12: Payload elements: If set to '`1`', payload elements will be part of the message
		Bit 0: DSP Status
		Bit 1: Gesture Data
		Bit 2: TouchInfo
		Bit 3: AirWheelInfo
		Bit 4: xyzPosition
		Bit 5: Noise Power
		Bit 6...10: These bits are reserved and must be set to '`0`'.
		Bit 11: CICData (Uncalibrated Signal)
		Bit 12: SDData (Signal Deviation)
		Bits 13...15: These bits are reserved and must be set to '`0`'
`Argument1`		Acts as a mask, set appropriate bits to '`1`' to change the flag. All other flags remain unchanged.

Note: For instances of the `Set_Runtime_Parameter` command examples, refer to Table A-2.

4.6 SENSOR_DATA_OUTPUT

This message contains the sensor data output of the GestIC. The content of the message can be configured via bit mask (refer to DataOutputEnableMask and DataOutputLockMask in **Section 4.5 "Set_Runtime_Parameter"**).

The elements DataOutputConfigMask, TimeStamp, and SystemInfo are always part of the message. The inclusion of further payload elements depends on the Configuration, and the actual Configuration can be read from the payload element DataOutputConfigMask.

Direction: GestIC to Host

TABLE 4-11: MESSAGE OVERVIEW

Header				Payload			
Size	Flags	Seq.	ID	DataOutputConfigMask	TimeStamp	SystemInfo	Variable depending on DataOutputConfigMask
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*	*1 Byte*	*1 Byte*	Variable depending on DataOutputConfigMask
variable	*n/a*	*n/a*	0x91	see description below			

TABLE 4-12: PAYLOAD ELEMENTS

Element	Element size (bytes)	Description
DataOutputConfigMask	2	Bit mask indicating which data is part of the message. The following bits are used: Bit 0: DSPStatus field. Bit 1: GestureInfo field. Bit 2: TouchInfo field. Bit 3: AirWheelInfo field. Bit 4: xyzPosition field. Bit 5: NoisePower field. Bit 6...10: These bits are reserved. Bit 11: CICData field. Bit 12: SDData field. Bit 13...15: These bits are reserved. ***Structure***: 16-bit Word read as a bit mask ***Range***: (0x0000...0xFFFF)
TimeStamp	1	8-Bit Counter of 200 Hz (Sample Interval) 200 Hz counter value wraps-around after 256 ticks. This indicates when an event has taken place and allows measuring the elapsed time between two events, as long as it is below approximately 1.25 seconds. ***Structure***: 8-bit Word read as decimal value. ***Range***: (0x00...0xFF)

TABLE 4-12: PAYLOAD ELEMENTS (CONTINUED)

Element	Element size (bytes)	Description
SystemInfo	1	Bit mask indicating if the respective sensor data is valid. In an application, the sensor data output should only be further processed if the respective bits are set to '1'. The following bits are used: Bit 0: PositionValid, if set indicates that the position in the xyzPosition field is valid. Bit 1: AirWheelValid, if set indicates that the AirWheel is active and the data in the AirWheelInfo field is valid. Bit 2: RawDataValid, if set indicates that the data of the CICData and SDData fields are valid; otherwise those fields must be ignored. Bit 3: NoisePowerValid, if set indicates that the NoisePower field is valid. Bit 4: EnvironmentalNoise, if set indicates that environmental noise has been detected. Bit 5: Clipping, if set indicates that the ADCs are clipping.(1) Bit 6: This bit is reserved. Bit 7: DSPRunning, if set indicates that the system is currently running. If not set, the system is about to go to Sleep. ***Structure***: 8-bit Word read as a bit mask ***Range***: (0x00...0xFF) **Note:** Position Data is disabled from the sensor data output and AirWheel is enabled: Position Valid will be set and sent with SystemInfo and a new message will be sent when AirWheel detection starts.
DSPStatus	2	This element consists of two bytes. The first byte contains information about Calibration events. The second byte indicates the Tx frequency currently used. Bit 0: This bit is reserved. Bit 1: CalibrationInfo: Forced Calibration (by Host) Bit 2: This bit is reserved. Bit 3: CalibrationInfo: Gesture triggered Bit 4: CalibrationInfo: Negative value Bit 5: CalibrationInfo: Idle Calibration Bit 6: CalibrationInfo: Invalid value Calibration Bit 7: CalibrationInfo: Calibration triggered by AFA Bits 8…15: Tx Frequency in kHz as decimal value (42...100) ***Structure***: 2 bytes; first byte is read as a bit mask while second as decimal ***Range***: (0x00...0xFF; 44...115) **Note:** TX frequency is only reported when the frequency changes.

Note 1: Clipping indication is not supported by MGC3140/MXG3141. Message is kept for compliance with MGC3x30.

TABLE 4-12: PAYLOAD ELEMENTS (CONTINUED)

Element	Element size (bytes)	Description
`GestureInfo`	4	This field contains the 32-bit gesture information Word. ***Recognized Gestures***: The recognized gestures are results of the HMM classification. Edge detection can be used to further classify where the gesture has been done (Edge Flicks). Furthermore, gesture attributes give information about the direction of the flick. The gesture information is given as a bit field and can be decoded as follows: Bits 0...7:Recognized gesture as decimal number 0: No gesture 1: Garbage model 2: Flick West to East 3: Flick East to West 4: Flick South to North 5: Flick North to South 6: Circle clockwise (only active if `AirWheel` disabled) 7: Circle counterclockwise (only active if `AirWheel` disabled) 8: Reserved 9: Reserved 64: Hold 65: Edge Flick West to East 66: Edge Flick East to West 67: Edge Flick South to North 68: Edge Flick North to South 69: Double Flick West to East 70: Double Flick East to West 71: Double Flick South to North 72: Double Flick North to South 73: Presence Bits 8...11: These bits must not be interpreted. Bits 12…15:Gesture Class read as a decimal number 0: Garbage model 1: Flick gesture 2: Circular gesture Bit 16: Edge flick – is '`1`' if flick gesture is classified as edge flick Bits 17...26:These bits are reserved. Bit 27: HandPresence flag: Is '`1`' while the user's hand is within the sensing space. Bit 28: HandHold flag: Is '`1`' while the hand is not moving. Further dependencies can be adjusted inside Aurea Parameterization. Bit 29: HandInside flag: Is '`1`' while the user's hand is approximately above the sensor. Bit 30: This bit is reserved. Bit 31: Gesture recognition in progress. This bit is set when the Gesture Recognizer is active and Reset when the gesture is recognized and the Recognizer is Off. ***Structure***: 32-bit Word read as a bit mask ***Range***: (`0x00000000...0xFFFFFFFF`)

TABLE 4-12: PAYLOAD ELEMENTS (CONTINUED)

Element	Element size (bytes)	Description
TouchInfo	4	Contains touch information The following bits are used to indicate a touch event on the respective electrodes: Bit 0: Touch South electrode Bit 1: Touch West electrode Bit 2: Touch North electrode Bit 3: Touch East electrode Bit 4: Touch Center electrode Bit 5: Tap South electrode Bit 6: Tap West electrode Bit 7: Tap North electrode Bit 8: Tap East electrode Bit 9: Tap Center electrode Bit 10: Double Tap South electrode Bit 11: Double Tap West electrode Bit 12: Double Tap North electrode Bit 13: Double Tap East electrode Bit 14: Double Tap Center electrode Bit 15: This bit is reserved. Bits 16...23:Touch Counter: 8-bit counter; this counter determines the period between the time when the hand starts moving to touch until it is detected. This period is equal to [Touch Counter Value] x 5 (ms). The counter starts counting when the minimum approach speed required to detect a touch event is exceeded, until the touch is detected. After each touch detection, the counter is reset. Bits 24...31:These bits are reserved. ***Structure***: 32-bit Word read as a bit mask ***Range***: (0x00000000...0xFFFFFFFF)
AirWheelInfo	2	The first byte contains a counter which indicates how far the AirWheel rotation has progressed. Bits 0...4: Value represents the current angular position with a resolution of 32 counts for a full revolution. Bits 5…7: Counts of full rotations. Each time the angular position crosses '0', a full revolution is counted. If the user's hand is moving in clockwise direction, the counter is increased. For counterclockwise movements, the counter is decreased. AirWheelInfo is only valid if the AirWheelValid bit in the element SystemInfo is '1'. The second byte is reserved. ***Structure***: Vector of two 8-bit Words read as a decimal value ***Range***: (0x0000...0x00FF)

TABLE 4-12: PAYLOAD ELEMENTS (CONTINUED)

Element	Element size (bytes)	Description
xyzPosition	6	This element contains x, y and z position data. Two bytes are used for each of the positions x, y and z. Bytes 1 and 2:x position Bytes 3 and 4:y position Bytes 5 and 6:z position The position information is only valid if the PositionValid bit in the element SystemInfo is '1'. The data give the position of the user's hand in the Cartesian coordinate system. Position data of [0,0,0] represent the origin of the coordinate system and data of [65535, 65535, 65535] are the maximum dimension of the sensing space. The origin is defined as the lower left corner of the sensitive space (South-West) at the surface of the system. ***Structure***: Vector of three16-bit Words read as decimal value for each position x, y, z ***Range***: (0x0000...0xFFFF) for each position x, y, z
NoisePower	4	Noise Power of the GestIC® system. NoisePower is only valid if the NoisePowerValid bit in the element SystemInfo is '1'. ***Structure***: 32-bit Word read as a float value ***Range***: (0...3.402823e+38)
CICData	20	Uncalibrated Sensor Data (CIC Data) ***Structure***: Vector of five, 32-bit Words interpreted as float values in format. An offset of 32000 needs to be added to each channel. xxxx.xxxx.xxxx.xxxx.xxxx (South.West.North.East.Center or S1.S2.S3.S4.North) format ***Range***: (-3.402823e+38...3.402823e+38) for each channel
SDData	20	Signal Deviation (SD) SDData are only valid if the RawDataValid bit in the element SystemInfo is '1'. ***Structure***: Vector of five, 32-bit Words interpreted as float values in xxxx.xxxx.xxxx.xxxx.xxxx (South.West.North.East.Center or S1.S2.S3.S4.North) format ***Range***: (-3.402823e+38...3.402823e+38) for each channel
Reserved	—	Reserved: Additional payload elements can be added in the future or for debug purposes.

Note: For the examples list of the Sensor_Data_Output command, refer to Table A-3.

Chapter 5. Messages for GestIC® Library Update

5.1 LIBRARY LOADER UPDATE PROCEDURE

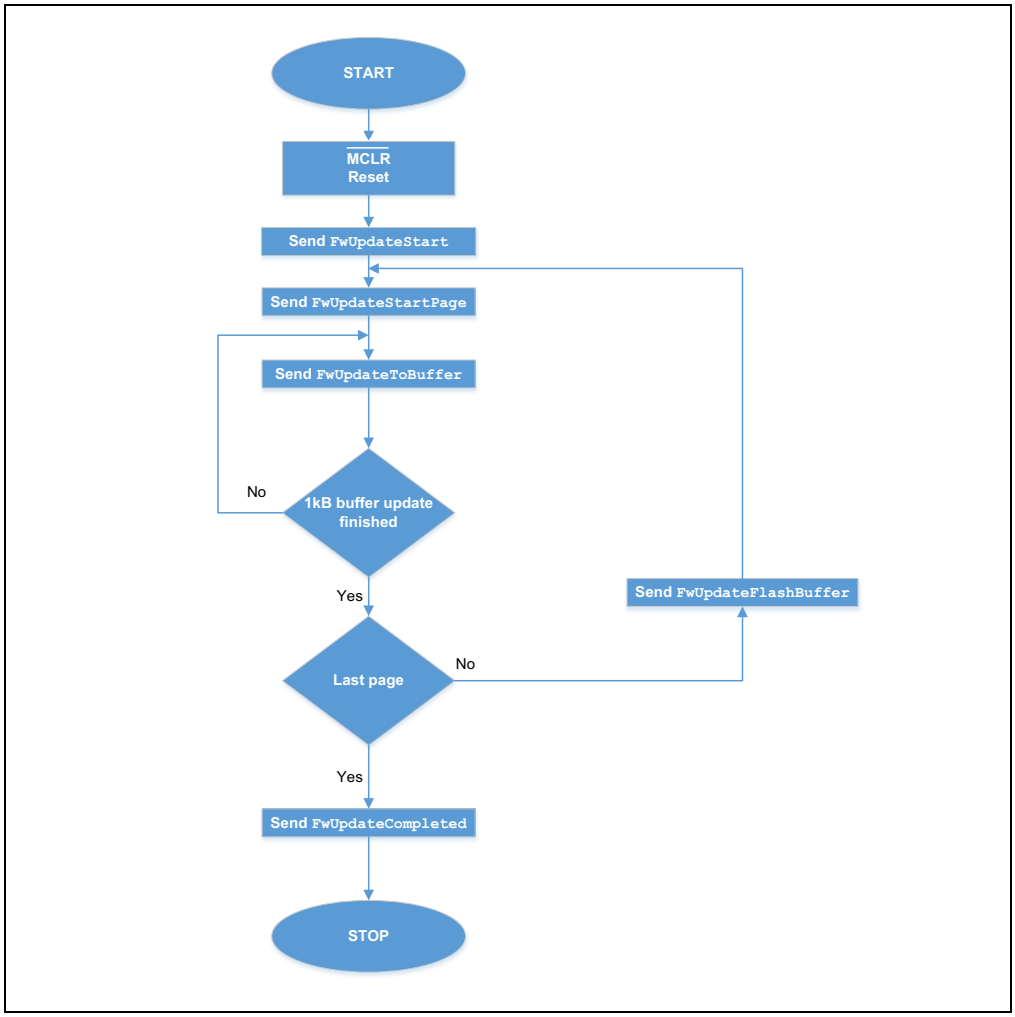
The general library update process is shown in Figure 5-1. Note that only libraries provided by Microchip Technology can be updated on the GestIC.

For the library update process, six different messages are used:

- `FwUpdateStart`
- `FwUpdateStartPage`
- `FwUpdateToBuffer`
- `FwUpdateFlashBuffer`
- `FwUpdateVerify`
- `FwUpdateCompleted`

Note: `FwUpdateStart` will erase all the library, so a partial update is not possible.

FIGURE 5-1: LIBRARY UPDATE FLOWCHART



5.2 FwUpdateStart

This message starts the update session of the GestIC device.

Direction: Host to GestIC

TABLE 5-1: MESSAGE OVERVIEW

Header				Payload					
Msg. Size	Flags	Seq.	ID	Crc	SessionId	FlashKey	UpdateFunction	ErasePageStart	ErasePageEnd
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*4 Bytes*	*8 Bytes*	*1 Byte*	*1 Byte*	*1 Byte*
0x17	0x00	*n/a*	0x70	**see description below**					

TABLE 5-2: PAYLOAD ELEMENTS

Field	Size (in bytes)	Description
Crc	4	CRC is calculated across the following fields: SessionId, FlashKey, UpdateFunction, ErasePageStart and ErasePageEnd. The same CRC calculation algorithm is used in the bootloader and firmware, with Ethernet polynomial 0x04C11DB7.
SessionId	4	The SessionId is a random number generated by the Host, to be sent via FwUpdateStart message, and resent via FwUpdateCompleted message. SessionId must be '0' if 'UpdateFunction = Restart', and different from '0' in all other cases.
FlashKey	8	FlashKey is the following sequence of bytes used to unlock Flash memory operations: {0x55, 0x66, 0x99, 0xAA, 0xAA, 0x99, 0x66, 0x55}.

UpdateFunction	1	The UpdateFunction is used here in ProgramFlash. The valid values are:
		
		`ProgramFlash = 0, // ProgramFlash`
		`VerifyOnly = 1, // VerifyOnly`
		`WaitForHostCommand = 2, // Wait for a host command`
		`Restart = 3 // Do a restart with Reset`

ErasePageStart	1	Must be '0'.
ErasePageEnd	1	Must be '0'.

Note: There are a total of 128 pages of program Flash memory. If 'UpdateFunction' is set to 'ProgramFlash', all of them will be erased to '0xFF' when message 'FwUpdateStart' is received by the device.

5.3 FwUpdateStartPage

This message stores the page number in an internal run-time variable, and initializes a 1K buffer with 0xFF.

Direction: Host to GestIC

TABLE 5-3: MESSAGE OVERVIEW

Header				Payload	
Msg. Size	Flags	Seq.	ID	Crc	PageNumber
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*1 Byte*
0x09	0x00	*n/a*	0x71	**see description below**	

TABLE 5-4: PAYLOAD ELEMENTS

Field	Size (bytes)	Description
Crc	4	CRC is calculated only for PageNumber
PageNumber	1	PageNumber indicates where to write in Flash memory with FwUpdateFlashBuffer message. The base memory address is given by PageNumber x 1024.

5.4 FwUpdateToBuffer

This message will copy the contents of Payload to the 1K buffer at Offset. The host must make sure that it does not write beyond the end of the 1K buffer. If the offset plus the length of data would go beyond the end of the buffer, a system status message with an error code will be sent by the bootloader, but the content will be written up until the end of the buffer. The length of Payload can be at maximum 128 bytes. It can be necessary to send more FwUpdateToBuffer messages to fill the 1K buffer with the required content.

Direction: Host to GestIC

TABLE 5-5: MESSAGE OVERVIEW

Header				Payload		
Msg. Size	Flags	Seq.	ID	Crc	Offset	Payload
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*2 Bytes*	*1...128 Bytes*
0x8A (max)	0x00	*n/a*	0x72	**see description below**		

TABLE 5-6: PAYLOAD ELEMENTS

Field	Size (bytes)	Description
Crc	4	CRC is calculated across Offset and Payload only.
Offset	2	Address Offset inside the buffer.
Payload	1...128	Data to store.

5.5 FwUpdateFlashBuffer

This message will flash the contents of the 1K buffer to Flash, but only if the SessionId is the same as was used with FwUpdateStart, the PageNumber is the same as used with the previous FwUpdateStartPage and less than the number of the last page, and the BufferCrc is correct for the current contents of the 1K buffer.

Direction: Host to GestIC

TABLE 5-7: MESSAGE OVERVIEW

Header				Payload				
Msg. Size	Flags	Seq.	ID	Crc	SessionId	BufferCrc	FlashKey	PageNumber
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*4 Bytes*	*4 Bytes*	*8 Bytes*	*1 Byte*
0x19	0x00	*n/a*	0x73	**see Table 5-8**				

TABLE 5-8: PAYLOAD ELEMENTS

Field	Size (bytes)	Description
Crc	4	CRC is calculated across the following fields: SessionId, BufferCrc, FlashKey and PageNumber
SessionId	4	The SessionId is the same random number as used for the FwUpdateStart. 0x00000000 is an invalid SessionId and is used to force the device into a restart.
BufferCrc	4	BufferCrc is the CRC calculated for all the 1K buffer using Ethernet polynominal 0x04C11DB7.
FlashKey	8	FlashKey is the following sequence of bytes used to unlock Flash memory operations: {0x55, 0x66, 0x99, 0xAA, 0xAA, 0x99, 0x66, 0x55}.
PageNumber	1	PageNumber is the same value used in the previous FwUpdateStartPage message.

5.6 FwUpdateVerify

This message has the same payload as FwUpdateFlashBuffer, but will not flash the data.

Instead it will just compare it with the flash contents at that page and output a system status message with a VerifyOk or a VerifyFailed status. If any more pages need to be updated, a FwUpdateStartPage is expected and the whole previous procedure repeated for the rest of the pages.

Unlike FwUpdateFlashBuffer, this message can be used for the last Flash page as well.

Direction: Host to GestIC

TABLE 5-9: MESSAGE OVERVIEW

Header				Payload				
Msg. Size	Flags	Seq.	ID	Crc	SessionId	BufferCrc	FlashKey	PageNumber
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*4 Bytes*	*4 Bytes*	*8 Bytes*	*1 Byte*
0x19	0x00	*n/a*	0x74	**see Table 5-10**				

TABLE 5-10: PAYLOAD ELEMENTS

Field	Size (bytes)	Description
Crc	4	CRC is calculated across the following fields: SessionId, BufferCrc, FlashKey and PageNumber
SessionId	4	The SessionId is the same random number as used for the FwUpdateStart. 0x00000000 is an invalid SessionId and is used to force the device into a restart.
BufferCrc	4	BufferCrc is the CRC calculated for all the 1K buffer using Ethernet polynominal 0x04C11DB7.
FlashKey	8	FlashKey is the following sequence of bytes used to unlock Flash memory operations: {0x55, 0x66, 0x99, 0xAA, 0xAA, 0x99, 0x66, 0x55}.
PageNumber	1	PageNumber is the same value used in the previous FwUpdateStartPage message.

5.7 FwUpdateCompleted

This message completes the update session of the GestIC.

Direction: Host to GestIC

TABLE 5-11: MESSAGE OVERVIEW

Header				Payload			
Msg. Size	Flags	Seq.	ID	Crc	SessionId	BufferCrc	FlashKey
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*4 Bytes*	*4 Bytes*	*4 Bytes*	*8 Bytes*
0x18	*n/a*	*n/a*	0x77	**see description below**			

TABLE 5-12: PAYLOAD ELEMENTS

Field	Size (in bytes)	Description
Crc	4	CRC is calculated across the SessionId, BufferCRC, and FlashKey fields.
SessionId	4	The SessionId is the same random number as used for the FwUpdateStart.
BufferCrc	4	BufferCrc is the CRC calculated for all the 1K buffer using Ethernet polynominal 0x04C11DB7.
FlashKey	8	FlashKey is the following sequence of bytes used to unlock Flash memory operations: {0x55, 0x66, 0x99, 0xAA, 0xAA, 0x99, 0x66, 0x55}.

Appendix A. I²C Command Examples

TABLE A-1: REQUEST_MESSAGE COMMAND EXAMPLES

Requested Function			Request Message												Comment
			Header				Payload								
			Msg. Size	Flags	Seq.	ID	Msg. ID	Reserved			Parameter				
—	FW version (0x83)		0x0C	0x00	0x00	0x06	0x83	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
Get Run-time Parameters	Electrode Mapping (0x0065, 0x0066, 0x0067, 0x0068, 0x0069)	Channelmapping_S	0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x65	0x00	0x00	0x00	Fixed command.
		Channelmapping_W	0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x66	0x00	0x00	0x00	
		Channelmapping_N	0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x67	0x00	0x00	0x00	
		Channelmapping_E	0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x68	0x00	0x00	0x00	
		Channelmapping_C	0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x69	0x00	0x00	0x00	
	Touch Detection (0x0097) and Approach Detection (0x0097)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x97	0x00	0x00	0x00	Fixed command.
	Approach Detection (0x0081)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x81	0x00	0x00	0x00	Fixed command.
	AirWheel (0x0090)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x90	0x00	0x00	0x00	Fixed command.
	Gesture Processing HMM (0x0085)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x85	0x00	0x00	0x00	Fixed command.
	Calibration Operation Mode (0x0080)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0x80	0x00	0x00	0x00	Fixed command.
	Data Output Enable Mask (0x00A0)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0xA0	0x00	0x00	0x00	Fixed command.
	Data Output Lock Mask (0x00A1)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0xA1	0x00	0x00	0x00	Fixed command.
	Data Output Request Mask (0x00A2)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0xA2	0x00	0x00	0x00	Fixed command.
	Gesture in Progress Flag Control (0x00A3)		0x0C	0x00	0x00	0x06	0xA2	0x00	0x00	0x00	0xA3	0x00	0x00	0x00	Fixed command.

TABLE A-2: `SET_RUNTIME_PARAMETER` COMMAND EXAMPLES

Requested Function			`Set_Runtime_Parameter`																Comment
			Header				Payload												
			Msg. Size	Flags	Seq.	ID	Runtime Parameter ID		Reserved		`Argument0`				`Argument1`				
Common Category	Trigger (0x1000)	Force Calibration	0x10	0x00	0x00	0xA2	0x00	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
		Enter Deep Sleep 1	0x10	0x00	0x00	0xA2	0x00	0x10	0x00	0x00	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
		Enter Deep Sleep 2	0x10	0x00	0x00	0xA2	0x00	0x10	0x00	0x00	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
	MakePersistent (0xFF00)	Store RTPs for AFE	0x10	0x00	0x00	0xA2	0x00	0xFF	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
		Store RTPs for DSP	0x10	0x00	0x00	0xA2	0x00	0xFF	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
		Store RTPs for System	0x10	0x00	0x00	0xA2	0x00	0xFF	0x00	0x00	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Fixed command.
Analog Front-End Category	Electrode Mapping (0x0065, 0x0066, 0x0067, 0x0068, 0x0069)	`Channelmapping_S`	0x10	0x00	0x00	0xA2	0x65	0x00	0x00	0x00	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00	`Argument0` (8-bit) defines the respective Rx Channel for each electrode. This value can be '`0`' for Rx0, '`1`' for Rx1, '`2`' for Rx2, '`3`' for Rx3 or '`4`' for Rx4. These values are just examples.
		`Channelmapping_W`	0x10	0x00	0x00	0xA2	0x66	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
		`Channelmapping_N`	0x10	0x00	0x00	0xA2	0x67	0x00	0x00	0x00	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
		`Channelmapping_E`	0x10	0x00	0x00	0xA2	0x68	0x00	0x00	0x00	0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
		`Channelmapping_C`	0x10	0x00	0x00	0xA2	0x69	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
Digital Signal Processing	TransFreqSelect (0x0082)	Five frequencies	0x10	0x00	0x00	0xA2	0x82	0x00	0x00	0x00	0x05	0x00	0x00	0x00	0x10	0x32	0x04	0x00	This is an example for five frequencies used in the following order (0x43210): ID 4, ID 3, ID2, ID 0
		Two frequencies	0x10	0x00	0x00	0xA2	0x82	0x00	0x00	0x00	0x02	0x00	0x00	0x00	0x42	0x00	0x00	0x00	This is an example for two frequencies used in the following order (0x42): ID 4, ID 2
	Touch Detection (0x0097)	Enable	0x10	0x00	0x00	0xA2	0x97	0x00	0x00	0x00	0x08	0x00	0x00	0x00	0x08	0x00	0x00	0x00	Fixed command.
		Disable	0x10	0x00	0x00	0xA2	0x97	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x08	0x00	0x00	0x00	Fixed command.
	Approach Detection (0x0097)	Enable	0x10	0x00	0x00	0xA2	0x97	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x01	0x00	0x00	0x00	Fixed command.
		Disable	0x10	0x00	0x00	0xA2	0x97	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	Fixed command.

TABLE A-2: `SET_RUNTIME_PARAMETER` COMMAND EXAMPLES (CONTINUED)

Requested Function			Set_Runtime_Parameter																Comment
			Header				Payload												
			Msg. Size	Flags	Seq.	ID	Runtime Parameter ID		Reserved		Argument0				Argument1				
System Category	AirWheel (0x0090)	Enable	0x10	0x00	0x00	0xA2	0x90	0x00	0x00	0x00	0x20	0x00	0x00	0x00	0x20	0x00	0x00	0x00	Fixed command.
		Disable	0x10	0x00	0x00	0xA2	0x90	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x20	0x00	0x00	0x00	Fixed command.
	Gesture Processing HMM (0x0085)	Enable All Gestures	0x10	0x00	0x00	0xA2	0x85	0x00	0x00	0x00	0x7F	0x00	0x00	0x00	0x7F	0x00	0x00	0x00	The `Argument0` (8-bit) defines which Gestures need to be configured. The `Argument1` defines the mask for the Gestures which need to be configured.These values are just examples.
		Enable Only Flick Gestures	0x10	0x00	0x00	0xA2	0x85	0x00	0x00	0x00	0x1F	0x00	0x00	0x00	0x7F	0x00	0x00	0x00	
		Enable in Addition Circles	0x10	0x00	0x00	0xA2	0x85	0x00	0x00	0x00	0x60	0x00	0x00	0x00	0x60	0x00	0x00	0x00	
	Calibration Operation Mode (0x0080)	Enable	0x10	0x00	0x00	0xA2	0x80	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x3F	0x00	0x00	0x00	Fixed command.
		Disable	0x10	0x00	0x00	0xA2	0x80	0x00	0x00	0x00	0x3F	0x00	0x00	0x00	0x3F	0x00	0x00	0x00	Fixed command.
	Data Output Enable Mask (0x00A0)	Enable All Data	0x10	0x00	0x00	0xA2	0xA0	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	0x3F	0x18	0x00	0x00	The `Argument0` defines which Data need to be enabled or disabled. The `Argument1` defines the mask for the Data which need to be configured.These values are just examples.
		Enable DSP, Gestures and Noise Power	0x10	0x00	0x00	0xA2	0xA0	0x00	0x00	0x00	0x23	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	
		Enable Only Data: Noise (others not changed)	0x10	0x00	0x00	0xA2	0xA0	0x00	0x00	0x00	0x10	0x00	0x00	0x00	0x10	0x00	0x00	0x00	
		Disable Only Data: CIC (others not changed)	0x10	0x00	0x00	0xA2	0xA0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x08	0x00	0x00	
	Data Output Lock Mask (0x00A1)	Lock All Data	0x10	0x00	0x00	0xA2	0xA1	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	0x3F	0x18	0x00	0x00	The `Argument0` defines which Data need to be locked or unlocked. The `Argument1` defines the mask for the Data which need to be configured.These values are just examples.
		Lock DSP, Gestures and Noise Power	0x10	0x00	0x00	0xA2	0xA1	0x00	0x00	0x00	0x23	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	
		Lock Only Data: Noise (others not changed)	0x10	0x00	0x00	0xA2	0xA1	0x00	0x00	0x00	0x10	0x00	0x00	0x00	0x10	0x00	0x00	0x00	
		UnLock Only Data: CIC (others not changed)	0x10	0x00	0x00	0xA2	0xA1	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x08	0x00	0x00	
	Data Output Request Mask (0x00A2)	Request All Data	0x10	0x00	0x00	0xA2	0xA2	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	0x3F	0x18	0x00	0x00	The `Argument0` defines which Data need to be requested. This is only valid for the next message. The `Argument1` defines the mask for the Data which need to be configured.These values are just examples.
		Request DSP, Gestures and Noise Power	0x10	0x00	0x00	0xA2	0xA2	0x00	0x00	0x00	0x23	0x00	0x00	0x00	0x3F	0x18	0x00	0x00	
		Request Only Data: Noise	0x10	0x00	0x00	0xA2	0xA2	0x00	0x00	0x00	0x10	0x00	0x00	0x00	0x10	0x00	0x00	0x00	
	Gesture in Progress Flag Control (0x00A3)	Enable	0x10	0x00	0x00	0xA2	0xA3	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x01	0x00	0x00	0x00	Fixed command.
		Disable	0x10	0x00	0x00	0xA2	0xA3	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	Fixed command.

TABLE A-3: `SENSOR_DATA_OUTPUT` COMMAND EXAMPLES

Requested Function	User Action	Sensor_Data_Output												Comment
		Header				Payload								
		Msg. Size	Flags	Seq.	ID	Data Output Config. Mask		Time Stamp	System Info	Parameter				
Data Output contains only `DSPStatus` field (configured using the `Set_Runtime_Parameter` command: `10 00 00 A2 A0 00 00 00 01 00 00 00 FF FF FF FF`)	No action	0x0A	0x08	0x26	0x91	0x01	0x01	0x5D	0x80	0x10	0x73	—	—	Negative Calibration.
		0x0A	0x08	0x27	0x91	0x01	0x01	0x5E	0x80	0x00	0x73	—	—	Calibration finished.
		0x0A	0x08	0x28	0x91	0x01	0x01	0x5D	0x80	0x20	0x73	—	—	Idle Calibration.
		0x0A	0x08	0x29	0x91	0x01	0x01	0x5E	0x80	0x00	0x73	—	—	Calibration finished.
Data Output contains only Gesture Data field (configured using the `Set_Runtime_Parameter` command: `10 00 00 A2 A0 00 00 00 02 00 00 00 FF FF FF FF`)	Flick East to west	0x0C	0x08	0x31	0x91	0x02	0x01	0x82	0x80	0x03	0x10	0x00	0x00	0x03: Flick East to West 0x10: Flick Gesture
		0x0C	0x08	0x32	0x91	0x02	0x01	0x83	0x80	0x00	0x00	0x00	0x00	
	Flick North to South	0x0C	0x08	0x33	0x91	0x02	0x01	0x13	0x80	0x05	0x10	0x04	0x00	0x05: Flick North to South 0x10: Flick Gesture
		0x0C	0x08	0x34	0x91	0x02	0x01	0x14	0x80	0x00	0x00	0x00	0x00	
	Flick South to North	0x0C	0x08	0x35	0x91	0x02	0x01	0x53	0x80	0x04	0x10	0x04	0x00	0x03: Flick South to North 0x10: Flick Gesture
		0x0C	0x08	0x36	0x91	0x02	0x01	0x54	0x80	0x00	0x00	0x00	0x00	
	Flick West to East	0x0C	0x08	0x37	0x91	0x02	0x01	0x5D	0x80	0x02	0x10	0x00	0x00	0x03: Flick West to East 0x10: Flick Gesture
		0x0C	0x08	0x38	0x91	0x02	0x01	0x5E	0x80	0x00	0x00	0x00	0x00	

TABLE A-3: `SENSOR_DATA_OUTPUT` COMMAND EXAMPLES (CONTINUED)

Requested Function	User Action	Sensor_Data_Output												Comment
		Header				Payload								
		Msg. Size	Flags	Seq.	ID	Data Output Config. Mask		Time Stamp	System Info	Parameter				
Data Output contains only Gesture Data field (configured using the `Set_Runtime_Parameter` command: `10 00 00 A2 A0 00 00 00 02 00 00 00 FF FF FF FF`) Gesture in Progress is activated using the `Set_Runtime_Parameter` command: `10 00 00 A2 A3 00 00 00 01 00 00 00 FF FF FF FF`)	Flick East to West	0x0C	0x08	0x3A	0x91	0x02	0x01	0x19	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x3B	0x91	0x02	0x01	0x45	0x81	0x03	0x10	0x00	0x00	Gesture recognized (Flick East to West)
		0x0C	0x08	0x3C	0x91	0x02	0x01	0x46	0x81	0x00	0x00	0x00	0x00	
	Just move hand	0x0C	0x08	0x3D	0x91	0x02	0x01	0x47	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x3E	0x91	0x02	0x01	0x6E	0x81	0x01	0x00	0x00	0x00	Garbage recognized
		0x0C	0x08	0x3F	0x91	0x02	0x01	0x6F	0x81	0x00	0x00	0x00	0x00	
	Flick East to West	0x0C	0x08	0x40	0x91	0x02	0x01	0x83	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x41	0x91	0x02	0x01	0xAC	0x80	0x03	0x10	0x04	0x00	Gesture recognized (Flick East to West)
		0x0C	0x08	0x42	0x91	0x02	0x01	0xAD	0x80	0x00	0x00	0x00	0x00	
	Flick North to South	0x0C	0x08	0x43	0x91	0x02	0x01	0x67	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x44	0x91	0x02	0x01	0x8A	0x80	0x05	0x10	0x04	0x00	Gesture recognized (Flick North to South)
		0x0C	0x08	0x45	0x91	0x02	0x01	0x8B	0x80	0x00	0x00	0x00	0x00	
	Flick South to North	0x0C	0x08	0x46	0x91	0x02	0x01	0x67	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x47	0x91	0x02	0x01	0x8E	0x80	0x04	0x10	0x04	0x00	Gesture recognized (Flick South to North)
		0x0C	0x08	0x48	0x91	0x02	0x01	0x8F	0x80	0x00	0x00	0x00	0x00	
	Flick West to East	0x0C	0x08	0x49	0x91	0x02	0x01	0x6E	0x81	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x4A	0x91	0x02	0x01	0x9A	0x80	0x02	0x10	0x02	0x00	Gesture recognized (Flick West to East)
		0x0C	0x08	0x4B	0x91	0x02	0x01	0x9B	0x80	0x00	0x00	0x00	0x00	
	Clockwise Circle	0x0C	0x08	0x4C	0x91	0x02	0x01	0x81	0x80	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x4D	0x91	0x02	0x01	0xD6	0x80	0x00	0x00	0x00	0x00	Circle Gesture not recognized because AirWheel is On
	Counter Clockwise Circle	0x0C	0x08	0x4E	0x91	0x02	0x01	0x05	0x80	0x00	0x00	0x00	0x80	Gesture Recognizer started
		0x0C	0x08	0x4F	0x91	0x02	0x01	0x56	0x80	0x00	0x00	0x00	0x00	Circle gesture not recognized because AirWheel is On

TABLE A-3: `SENSOR_DATA_OUTPUT` COMMAND EXAMPLES (CONTINUED)

Requested Function	User Action	Sensor_Data_Output												Comment
		Header				Payload								
		Msg. Size	Flags	Seq.	ID	Data Output Config. Mask	Time Stamp	System Info	Parameter					
Data Output contains only Touch Data field (configured using the `Set_Runtime_Parameter` command: `10 00 00 A2 A0 00 00 00 04 00 00 00 FF FF FF FF`)	Touch Center Electrode	0x0C	0x08	0x45	0x91	0x04	0x01	0x51	0x81	0x10	0x00	0x09	0x00	Center Touch detected and the touch counter = 0x09
		0x0C	0x08	0x46	0x91	0x04	0x01	0x52	0x81	0x10	0x00	0x00	0x00	Touch Counter Reset
		0x0C	0x08	0x47	0x91	0x04	0x01	0x5D	0x81	0x00	0x02	0x00	0x00	Tap on Center electrode detected
		0x0C	0x08	0x48	0x91	0x04	0x01	0x5E	0x81	0x00	0x00	0x00	0x00	

Appendix B. Glossary

TABLE B-1: GLOSSARY

Term	Definition
AFE	Analog front-end
Application Host	PC or embedded controller which controls the GestIC®
Aurea	GestIC® PC control software with graphical user interface
Colibri Suite	Embedded DSP suite within the GestIC® Library
Deep Sleep	GestIC® Power-Saving mode
E-field	Electrical field
Frame Electrodes	Rectangular set of four electrodes for E-field sensing
GestIC® Technology	Microchip's patented technology providing 3D free-space gesture recognition utilizing the principles of electrical near-field sensing
GestIC® Library	Includes the implementation of GestIC® features and is delivered as a binary file preprogrammed on the GestIC®
Gesture Recognition	Microchip's stochastic HMM classifier to automatically detect and classify hand movement patterns
Gesture Set	A set of provided hand movement patterns
Hand Brick	Copper-coated test block (40x40x70 mm)
Hillstar	MGC3130 Development Kit
HMM	Hidden Markov Model
Position Tracking	GestIC® technology feature
Sabrewing	MGC3x30 evaluation board
Self Wake-up	MGC3140 Power-Saving mode
Sensing Area	Area enclosed by the four-frame electrodes
Sensing Space	Space above sensing area
Signal Deviation	Term for the delta of the sensor signal on approach of the hand versus non-approach
Spacer Brick	Spacer between the sensor layer and hand brick (Styrofoam block 40x40xh mm) with h = 1/2/3/5/8/12 cm
SPU	Signal Processing Unit
Approach Detection	GestIC® technology feature: Power-Saving mode of the MGC3140 with approach detection (not supported by MXG3141)
Woodstar	MGC3030 Development Kit

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820