# MGC3130 GestIC® Library Interface Description User's Guide

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "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.



**Trademarks**



Printed on recycled paper.

ISBN: 9781620776490

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

# 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 web site (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 on-line help. Select the Help menu, and then Topics to open a list of available online help files.**

## INTRODUCTION

This chapter contains general information that will be useful to know before using the MGC3130 GestIC® Library Interface. Items discussed in this chapter include:

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

## DOCUMENT LAYOUT

This document describes the MGC3130 GestIC Library and is organized as follows:

- **Chapter 1. Introduction**
- **Chapter 2. MGC3130 Host Interface**
- **Chapter 3. GestIC Library Message Interface**
- **Chapter 4. GestIC Library Message Reference**
- **Chapter 5. Messages for GestIC Library Update**

## 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 web site.

## RECOMMENDED READING

This user's guide describes how to use MGC3130 GestIC Library Interface. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

- *"MGC3130 Single-Zone 3D Gesture Controller Data Sheet"* (DS40001667) – Consult this document for information regarding the MGC3130 3D Tracking and Gesture Controller.
- *"MGC3130 Aurea Graphical User Interface User's Guide"* (DS40001681) – Describes how to use the MGC3130 Aurea Graphical User Interface.
- *"MGC3130 GestIC® Design Guide"* (DS40001716) – This document describes the MGC3130 system characteristic parameters and the design process. It enables the user to generate a good electrode design and to parameterize the full GestIC system.

## THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Information about GestIC technology and MGC3130 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 web site 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 web site at:

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

## DOCUMENT REVISION HISTORY

### Revision A (August 2013)

- Initial release of the document.

### Revision B (November 2013)

- Updated Chapters 1, 2, 3 and 4; Added Chapter 5; Updated content for GestIC Library V1.0 and later.

# Chapter 1. Introduction

## 1.1 PURPOSE OF THIS DOCUMENT

This document is the interface description of the MGC3130's GestIC Library. It outlines the function of the Library's I²C™ message interface, and contains the complete message reference to control and operate the MGC3130 system.

The main sections covered are:

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

The parameterization of the Colibri Suite is not covered in this document. That is only possible via Aurea PC software. Please refer to *"MGC3130 Aurea Graphical User Interface"* (DS40001681).

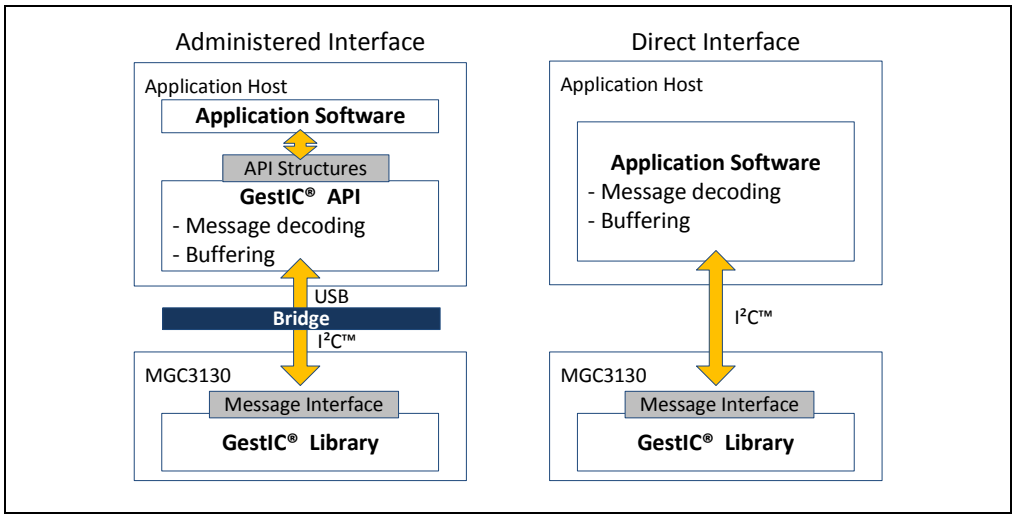
## 1.2 MGC3130 SOFTWARE ARCHITECTURE

A MGC3130 system can be accessed at two software levels:

- by direct I²C access via message interface of GestIC Library (direct interface)
- by GestIC API as an abstraction layer of the messages (administered interface)

Examples for the two principal options are shown in Figure 1-1.

**FIGURE 1-1: EXAMPLES FOR MGC3130 SOFTWARE ACCESS**




The direct interface is the simplest way to access MGC3130, but it requires the user to receive and decode all I²C messages and validate received data. Direct access is recommended if a reduced set of sensor data are used by the application (e.g., gestures only, position only). The administered interface via GestIC API provides decoded and validated sensor data, which can be immediately used in the application. Typically, GestIC API runs in PC applications or OS drivers, which provide data to the application software.

The following sections give a brief description of the building blocks of the two interface modes.

## 1.3 GestIC® LIBRARY

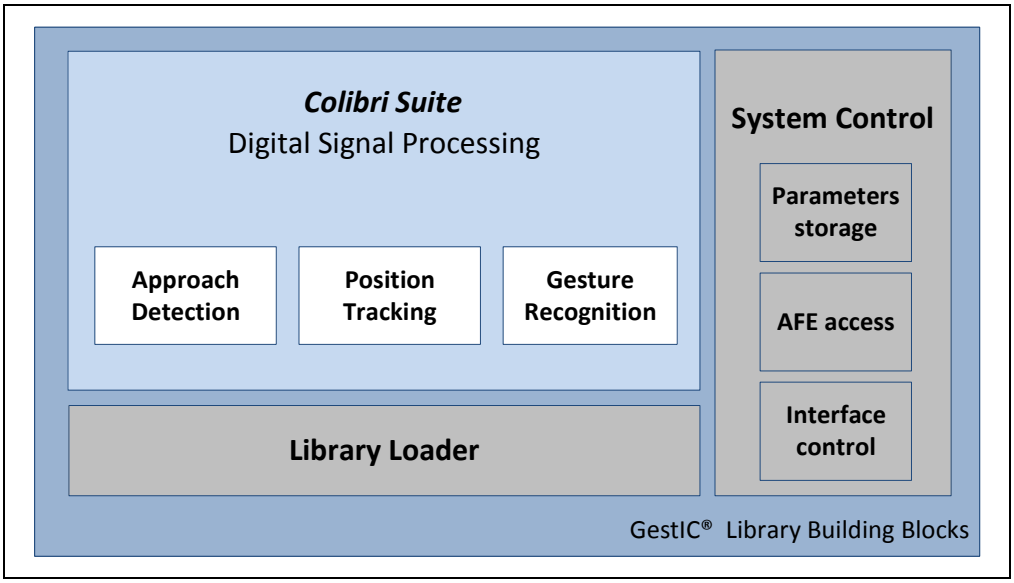
The GestIC Library is embedded firmware stored on the MGC3130's internal Flash memory. It contains:

- 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 providing full control of host interfaces, parameter storage and AFE access
- the Library Loader for updates of 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 sensor data to the host application.

**FIGURE 1-2: BUILDING BLOCKS OF GestIC® LIBRARY**




## 1.4 BRIDGE

An additional hardware bridge is needed if the application host does not support a native $I^2C$ interface. The bridge converts the $I^2C$ hardware protocol to USB/UART.

If a bridge hardware is incorporated, the application host may need an additional device driver to register the interface and provide MGC3130 data within the operating system.

Examples are:

- Windows® CDC driver to send MGC3130 data to a virtual COM port. In this case, the driver is not aware of the MGC3130 data format.
- HID driver to use the MGC3130 data directly as USB HID classes within the operating system. Such driver must decode MGC3130 messages and, thus, the GestIC API reference code is recommended to be part of it.

## 1.5 GestIC API

As an abstraction layer for MGC3130 messages, Microchip developed the GestIC API to provide a simplified user interface which can be easily integrated into the customer's application.

GestIC API comes along with a C reference code which includes message buffer, decoder and event handler to make the interface independent from the low-level protocol and its timing constraints.

## 1.6 APPLICATION SOFTWARE

The sensor output is used in a user's application which integrates context-driven actions based on the user's hand movements.

Typically, the application software provides a graphical user interface (GUI) to visualize the MGC3130 control options, like Aurea, which is delivered within the MGC3130 evaluation and development kits.

# Chapter 2. MGC3130 Host Interface

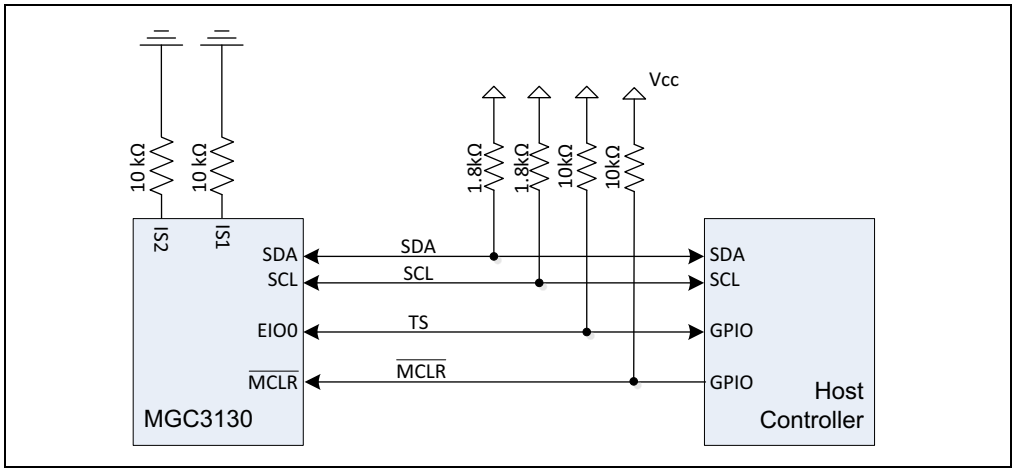
## 2.1 MGC3130 HARDWARE INTERFACE

Communication with the MGC3130 is accomplished via a two-wire I²C compatible serial port, which allows the user to read the sensor data and to send control messages to the chip. It communicates via the serial interface with a master controller, which operates at speeds up to 400 kHz. One pin (IS2) is available for address selection and enables the user to connect up to two MGC3130 devices on the same bus without address conflict.

> **Note:** The MGC3130 I²C™ addresses are 0x42 and 0x43. They are given as device addresses without the R/$\overline{W}$ bit. Please compare to the *"MGC3130 Single-Zone 3D Gesture Controller Data Sheet"* (DS40001667).

In addition, MGC3130 requires a dedicated transfer status line (TS), which features a data transfer status function. It is used by both I²C Master and Slave to control the data flow. I²C SCL, I²C SDA and TS lines require an open-drain connection on MGC3130 and the connected host controller. To function properly, I²C SCL and I²C SDA need to be pulled up to VCC with 1.8 kΩ resistors and the TS line needs to be pulled up to VCC with a 10 kΩ resistor.

**FIGURE 2-1: HARDWARE INTERFACE TO HOST CONTROLLER**




In order to complete the control options for MGC3130, it is recommended that the host controller controls the MGC3130 $\overline{MCLR}$ line. In particular, the hardware reset is necessary for the update procedure of the GestIC Library.

## 2.2 USAGE OF TRANSFER STATUS LINE (TS)

The transfer status line is used to check if $I^2C$ data are valid and if they can be sent from MGC3130 to the host controller.

The MGC3130 ($I^2C$ Slave) uses this line to inform the host controller ($I^2C$ Master) that there is data available which can be transferred. The host controller uses the TS line to indicate that data are being transferred and prevents MGC3130 from updating its data buffer.

Table 2-1 shows how the TS line is used in the different states of communication.

MGC3130 can update the $I^2C$ buffer only when TS is released by both chips, and a data transfer can only be started when MGC3130 pulls TS low.

This procedure secures that:

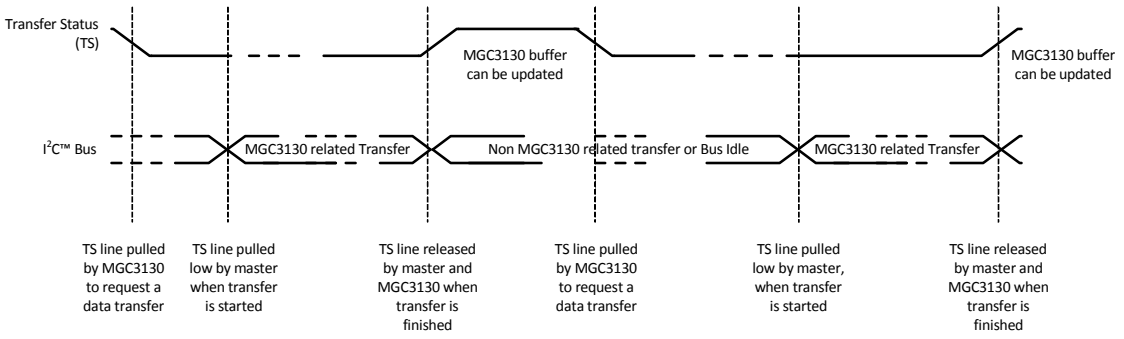
- the host is always informed when new sensor data are available
- buffer updates in MGC3130 are always completed before data are sent to the $I^2C$ bus

Figure 2-2 shows the complete communication protocol.

**TABLE 2-1: USAGE OF TRANSFER STATUS LINE**

| MGC3130 | Host Controller | TS Line | Status |
|---|---|---|---|
| Released (H) | Released (H) | High | Host finished reading data (Transfer end). No more data to be transferred to the host. MGC3130 is allowed to update the data buffer. |
| Asserted (L) | Released (H) | Low | Data from MGC3130 is available to be sent, but the host has not yet started reading. If the host is busy and did not start reading before the next data update (5 ms), the MGC3130 will assert the TS line high while updating the data buffer. |
| Asserted (L) | Asserted (L) | Low | Host starts reading. MGC3130 data buffer will not be updated until the end of transfer (host releases TS high). |
| Released (H) | Asserted (L) | Low | MGC3130 is ready to update the data buffer, but the host is still reading the previous data. MGC3130 is allowed to update the data only when the host releases the TS high. |

**FIGURE 2-2: MGC3130 COMMUNICATION PROTOCOL**




**Note 1:** The Stop condition after an I²C™ data transmission is generated by the host controller (I²C™ Master) after the data transfer is completed. Thus, it is recommended to verify the amount of bytes to be read in the message header (Size field).

**2:** Transfer Status is only needed for data transfer from MGC3130 to the host controller. Writing to MGC3130 does not require the additional TS signal.

## 2.3 CODING EXAMPLE

In addition to the standard I²C interface, the communication between MGC3130 and the host controller requires a proper handling of the Transfer Status. For an easier integration, the requirements are put into the code examples below.

**EXAMPLE 2-1: CODE IMPLEMENTATION IN HOST CONTROLLER**

```
I²C™ Read Function - requires TS:
I²C™ Master read loop:
    Read TS
    If TS == 0:
       Assert TS
       Send I²C™ start condition
       Send I²C™ device address + read indication
       Receive I²C™ payload (the GestIC® Library message)
       Send I²C™ stop condition
       Release TS
    Wait 200 µs (to assure that MGC3130 released TS line, too)
I²C™ Write Function - does not require TS:
I²C™ Master write loop:
       Send I²C™ start condition
       Send I²C™ device address + write indication
       Send I²C™ payload (the GestIC® Library message)
       Send I²C™ stop condition
```

# 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 MGC3130 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 |
|---|---|---|
| 0x15 | System_Status | 25 |
| 0x06 | Request_Message | 27 |
| 0x83 | Fw_Version_Info | 28 |
| 0xA2 | Set_Runtime_Parameter | 29 |

**TABLE 3-2: MESSAGE FOR SENSOR DATA OUTPUT**

| ID | Name | Page |
|---|---|---|
| 0x91 | Sensor_Data_Output | 35 |

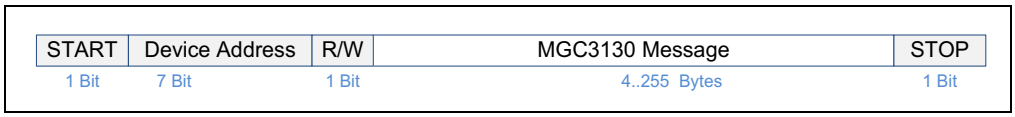
**TABLE 3-3: MESSAGES FOR GestIC® LIBRARY UPDATE**

| ID | Name | Page |
|---|---|---|
| 0x80 | Fw_Update_Start | 39 |
| 0x81 | Fw_Update_Block | 40 |
| 0x82 | Fw_Update_Completed | 42 |

## 3.2 MESSAGE FORMAT

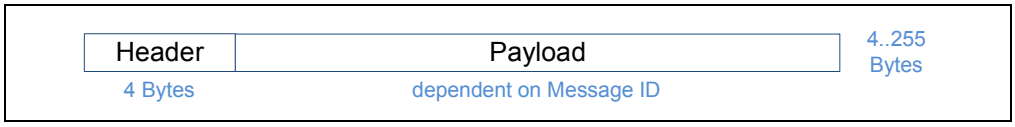
A message is the container to exchange data between GestIC Library and the application host. Each message has a length minimum of 4 bytes and a maximum of 255 bytes, and fits into the data packets of the communication interface (e.g., $I^2C$). Each frame transports a single message (see Figure 3-1).

**FIGURE 3-1: MGC3130 MESSAGE EMBEDDED IN THE $I^2C^{TM}$ FRAME**




Messages consist always of a 4-byte header and a variable payload. The format is shown in Figure 3-2.

**FIGURE 3-2: MGC3130 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 shown in Figure 3-3 and explained in Table 3-4.

**FIGURE 3-3: MGC3130 MESSAGE HEADER**

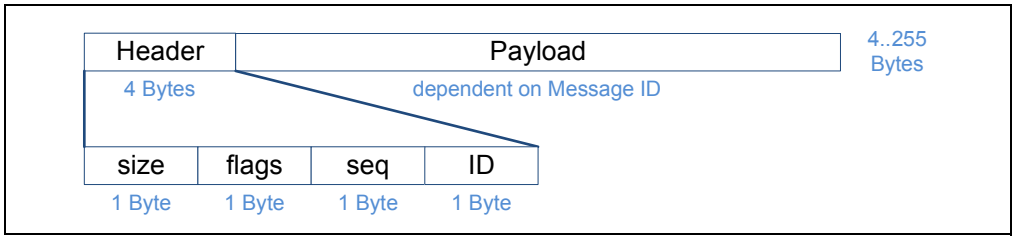


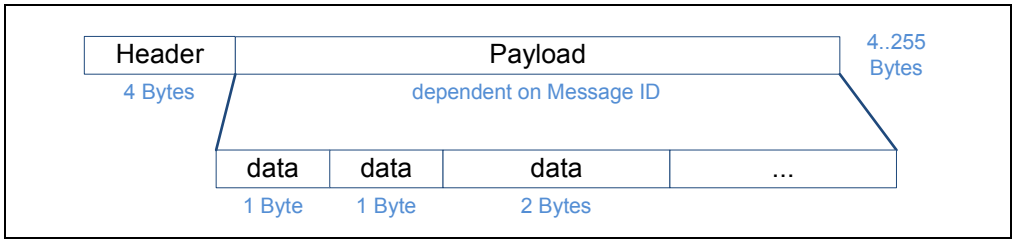

**TABLE 3-4: DATA FIELDS OF MGC3130 MESSAGE HEADER**

| 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 MGC3130. Range is 0…255. The host controller can use that information to verify if the messages got lost during $I^2C$™ transmission. MGC3130 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: MGC3130 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 `0x83, FW_Version_Info` is decoded in the following section.

**EXAMPLE 3-1: HEXADECIMAL REPRESENTATION OF MESSAGE `0x83`**

```
84 00 00 83 AA 63 80 E6 0C 64 15 20 31 2E 30 2E 30 3B 70 3A 48 69 6C 6C 73 74 61 72
56 30 31 3B 44 53 50 3A 49 44 39 30 30 30 72 31 38 34 39 3B 69 3A 42 3B 66 3A 32 32
35 30 30 3B 6E 4D 73 67 3B 73 3A 42 65 74 61 32 72 31 30 34 30 3A 31 30 34 39 3A 4D
4F 3B 63 3A 4D 4B 49 3B 74 3A 32 30 31 33 2F 31 31 2F 30 38 20 31 33 3A 30 33 3A 30
00 10 00 00 55 AA 90 65 20 20 80 0F FF 00 FF 00 E1 EA 00 00
```

### 3.5.1 Header Extraction

**EXAMPLE 3-2: MESSAGE HEADER**

```
84 00 00 83 AA 63 80 E6 0C 64 15 20 31 2E 30 2E 30 3B 70 3A 48 69 6C 6C 73 74 61 72
56 30 31 3B 44 53 50 3A 49 44 39 30 30 30 72 31 38 34 39 3B 69 3A 42 3B 66 3A 32 32
35 30 30 3B 6E 4D 73 67 3B 73 3A 42 65 74 61 32 72 31 30 34 30 3A 31 30 34 39 3A 4D
4F 3B 63 3A 4D 4B 49 3B 74 3A 32 30 31 33 2F 31 31 2F 30 38 20 31 33 3A 30 33 3A 30
00 10 00 00 55 AA 90 65 20 20 80 0F FF 00 FF 00 E1 EA 00 00
```

The message header contains the following information:

- *Size:* `0x84` *Message including header is 132 bytes long*
- *Flags:* `0x00` *Flags are not set*
- *Seq.:* `0x00` *The message has been sent out with a sequence number of 0*
- *ID:* `0x83` *The message ID is* `0x83, Fw_Version_Info`

### 3.5.2 Payload Extraction

**EXAMPLE 3-3: MESSAGE PAYLOAD**

```
84 00 00 83 AA 63 80 E6 0C 64 15 20 31 2E 30 2E 30 3B 70 3A 48 69 6C 6C 73 74 61 72 56
30 31 3B 44 53 50 3A 49 44 39 30 30 30 72 31 38 34 39 3B 69 3A 42 3B 66 3A 32 32 35 30
30 3B 6E 4D 73 67 3B 73 3A 42 65 74 61 32 72 31 30 34 30 3A 31 30 34 39 3A 4D 4F 3B 63
3A 4D 4B 49 3B 74 3A 32 30 31 33 2F 31 31 2F 30 38 20 31 33 3A 30 33 3A 30 00 10 00 00
55 AA 90 65 20 20 80 0F FF 00 FF 00 E1 EA 00 00
```

According to **Section 4.3 "Fw_Version_Info"**, `Fw_Version_Info` holds five payload elements:

- **FwValid** Status of GestIC Library (1 byte)
- **HwRev** HW revision information (2 bytes)
- **ParameterStartAddr** Start address of parameter (1 byte)
- **LibaryLoaderVersion** GestIC Library loader version (3 bytes)
- **FwStartAddr** Start address of GestIC Library(1 byte)
- **FwVersion** Version information of GestIC Library if valid (120 bytes)

The values can now be converted and mapped to the description of the payload elements:

**FwValid** = `AA (170)`: A valid GestIC Libary is available

**HwRev** = `63 80` (read as `0xE6 0x80`): HW revision is `230.128`

**ParameterStartAddr** = `0xE6` (`230x128=29440`) Start address of parameter is 29440

**LibaryLoaderVersion** = `0C 64 15` (read as `0x15 0x64 0x0C`): Library Loader version is `21.100.12`

**FwStartAddr** = `0x20 (32x128=4096)`: Start address of GestIC Library is 4096

**FwVersion** = `31 2E 30 2E 30 3B 70 3A 48 69 6C 6C 73 74 61 72 56 30 31 3B 44 53 50 3A 49 44 39 30 30 30 72 31 38 34 39 3B 69 3A 42 3B 66 3A 32 32 35 30 30 3B 6E 4D 73 67 3B 73 3A 42 65 74 61 32 72 31 30 34 30 3A 31 30 34 39 3A 4D 4F 3B 63 3A 4D 4B 49 3B 74 3A 32 30 31 33 2F 31 31 2F 30 38 20 31 33 3A 30 33 3A 30 00 10 00 00 55 AA 90 65 20 20 80 0F FF 00 FF 00 E1 EA 00 00`

The version string is interpreted by ASCII characters. It is a semicolon-separated string, always starting with the Version Number itself, followed by different tags:

```
1.0.0;p:HillstarV01;DSP:ID9000r1849;i:B;f:22500;nMsg;s:Beta2r10
40:1049:MO;c:MKI;t:2013/11/08 13:03:0;...
```

## 3.6 MESSAGE CONTROL FLOW AND CODING EXAMPLES

### 3.6.1 Message Control Flow

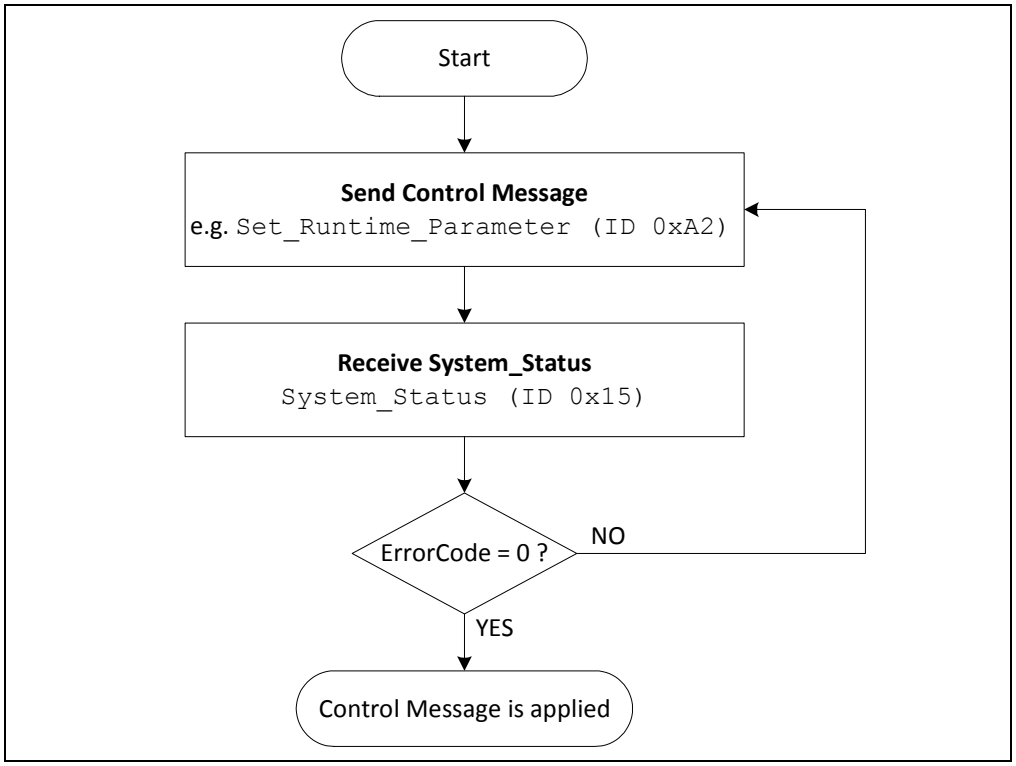
The control of MGC3130 GestIC Library is done through the following messages:

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

MGC3130 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 MGC3130.

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 Hillstar I²C™ to USB bridge prefixes every I²C™ packet with `0xFEFF` 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 MGC3130 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, MGC3130 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 runtime 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 MGC3130: `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. |

MGC3130 replies with message `FW_Version_Info (0x83)` followed by `System_Status (0x15)`, containing the error code.

**TABLE 3-6: MESSAGE FROM MGC3130 TO HOST: `FW_VERSION_INFO (0X83)`**

| **Raw Message** | 84 00 01 83 AA 00 00 FF 00 00 00 20 31 2E 30 2E 30 3B 70 3A 48 69 6C 6C 73 74 61 72 56 30 31 3B 44 53 50 3A 49 44 39 30 30 30 72 31 38 34 39 3B 69 3A 42 3B 66 3A 32 32 35 30 30 3B 6E 4D 73 67 3B 73 3A 42 65 74 61 32 72 31 30 34 30 3A 31 30 34 39 3A 4D 4F 3B 63 3A 4D 4B 49 3B 74 3A 32 30 31 33 2F 31 31 2F 30 38 20 31 33 3A 30 33 3A 30 38 3B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E1 EA 00 00 | | | | | |
|---|---|---|---|---|---|---|
| Payload Element | FWValid | HWRev | Parameter-StartAddr | LibraryLoad-erVersion | FWStartAddr | FWVersion |
| Hex in little endian | AA | 00 00 | FF | 00 00 00 | 20 | ... |
| Hex decoded | 0xAA | n.a. | n.a. | n.a. | 0x20 | ... |
| Description | 170 ValidFW | Only valid after MGC3130 start-up | Only valid after MGC3130 start-up | Only valid after MGC3130 start-up | Start address of GestIC® Library | Please see below |

`FWVersion` interpreted as ASCII characters:

```
1.0.0;p:HillstarV01;DSP:ID9000r1849;i:B;f:22500;nMsg;s:Beta2r10
40:1049:MO;c:MKI;t:2013/11/08 13:03:08;…
```

- GestIC Library Version: 1.0.0
- Plattform: HillstarV01
- Colibri Suite Version: ID9000r1849
- Build Time: 2013/11/08 13:03:08

### 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 MGC3130. For a detailed message description, please refer to **Section 4.4 "Set_Runtime_Parameter"**.

The following examples show how to set relevant runtime 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)`.

**TABLE 3-7: MESSAGE FROM HOST TO MGC3130: 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 | x00000001 |
| Description | FW_Version_Info | n.a. | | Enable Approach Detection mode | Mask for Approach Detection bit |

MGC3130 replies with message `System_Status (0x15)`, containing the error code.

**TABLE 3-8: MESSAGE FROM MGC3130 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 | InfoField1 | InfoField2 |
| 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 MGC3130: SET_RUNTIME_PARAMETER (0XA2)**

| Raw Message | 10 00 00 A2 | 85 00 | 00 00 | 7F 00 00 00 | 00 00 00 00 |
|---|---|---|---|---|---|
| Payload Element | RuntimeParameterID | Reserved | | Argument0 | Argument1 |
| Hex in little endian | 85 00 | 00 00 | | 7F 00 00 00 | 00 00 00 00 |
| Hex decoded | 0x0085 | n.a. | | 0x0000007F | n.a. |
| Description | despGestureMask | n.a. | | Enable gestures 0...6 | Argument1 is not used |

MGC3130 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. Please refer to **Section 4.4.11 "Data Output Enable Mask"**.

**TABLE 3-10: MESSAGE FROM HOST TO MGC3130: 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 | 0xA0 | 0x0000 | 0x0000001E | 0xFFFFFFFF |
| Description | DataOutputEn-ableMask | n.a. | Enable bit 1...bit 4; disable all other bits | Overwrite existing configuration |

MGC3130 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. Please refer to **Section 4.4.12 "Data Output Lock Mask"**.

**TABLE 3-11: MESSAGE FROM HOST TO MGC3130: 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 |

MGC3130 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 $I^2C$ message buffer is regularly updated in that time interval. Whenever new data are available, MGC3130 pulls the TS line to request the $I^2C$ master to transfer this data. Sensor data sent from MGC3130 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 MGC3130 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<br>Bit 3: NoisePowerValid<br>Bit 7: DSPRunning | Flick East to West | No touch | No AirWheel | No Position Data available |

**TABLE 3-13: MESSAGE FROM MGC3130 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<br>Byte 3 and 4: 0x5312<br>Byte 5 and 6: 0x0A6B |
| Description | Bit 0: PositionValid<br>Bit 2: RawDataValid<br>Bit 3: NoisePowerValid<br>Bit 7: DSPRunning | No Gesture Detected | Touch on Center Electrode | No AirWheel Data | x: 42586<br>y: 21266<br>z: 2667 |

**TABLE 3-14: MESSAGE FROM MGC3130 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<br>Byte 3 and 4: 0x87E7<br>Byte 5 and 6: 0x356a |
| Description | Bit 0: PositionValid<br>Bit 2: RawDataValid<br>Bit 3: NoisePowerValid<br>Bit 7: DSPRunning | No Gesture Detected | Touch on Center Electrode | No AirWheel Data | x: 45615<br>y: 34791<br>z: 13674 |

# Chapter 4. GestIC Library Message Reference

## 4.1 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 (0x06)
- Set_Runtime_Parameter (0xA2)
- Fw_Update_Start (0x80)
- Fw_Update_Block (0x81)
- Fw_Update_Completed (0x82)

Direction: MGC3130 to Host

**TABLE 4-1: MESSAGE OVERVIEW**

| Header | | | | Payload | | | | |
|---|---|---|---|---|---|---|---|---|
| Msg. Size | Flags | Seq. | ID | MsgID | MaxCmdSize | ErrorCode | InfoField1 | InfoField2 |
| *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *2 Bytes* | *4 Bytes* | *4 Bytes* |
| 16 | n.a. | n.a. | 0x15 | see description below | | | | |

**TABLE 4-2: PAYLOAD ELEMENTS**

| Element | Element Size (in bytes) | Description |
|---|---|---|
| MsgID | 1 | Holds the Message ID which System_Status corresponds to<br>***Structure***: 1 byte<br>***Range***: (0..0xFF) |
| MaxCmdSize | 1 | Holds the maximum I²C™ packet size GestIC® Library accepts (including header)<br>***Structure***: 1 byte<br>***Range***: (0..0xFF) |
| ErrorCode | 2 | Error code, returned for the previous message.<br>***Structure***: 16-bit word containing dedicated values (see list below)<br>Possible values:<br>0 NoError — o.k.<br>1 UnknownCommand — MessageId is unknown<br>2 InvalidSessionId — Session ID is invalid or does not match (0x0 is not allowed) (message FwUpdateStart, FwUpdateCompleted)<br>3 InvalidCrc — CRC is invalid thrown by commands: FwUpdateBlock, FwUpdateStart,FwUpdateCompleted<br>4 InvalidLength — Length is invalid (message FwUpdateBlock)<br>5 InvalidAddress — Address is invalid (message FwUpdateBlock)<br>6 InvalidFunction — Function-id is invalid (message FwUpdateStart, FwUpdateBlock)<br>8 ContentMismatch — The VerifyOnly function found a mismatch between content and Flash memory (message: FwUpdateBlock)<br>9 NoClientReachable — A client is not available, or communication is lost<br>10 NoFwPresent — No valid FW is present to execute<br>11 WrongParameterAddr — Parameter address does not match Bootloader assumption<br>20 WrongParameterValue — The value of the parameter is out of the valid range<br>21 UnknownParameterID — An unknown parameter ID is given<br>25 PersistentDataVersionMismatch — Persistent Data version is not correct<br>26 WakeupHappend — A wake-up by Host was detected<br>27 TrimValuesCorrupt — Trim values are invalid or corrupt<br>65280 RuntimeError — Runtime error occurred. |
| InfoField1 | 4 | Reserved |
| InfoField2 | 4 | Reserved |

## 4.2 REQUEST_MESSAGE

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

Direction: Host to MGC3130

**TABLE 4-3: 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* |
| 12 | n.a. | n.a. | 0x06 | see description below | | |

**TABLE 4-4: PAYLOAD ELEMENTS**

| Element | Element Size (in bytes) | Description |
|---|---|---|
| MessageID | 1 | Request the Message with ID MessageID from GestIC® Library. GestIC® Library shall answer with the requested message or stay silent.<br>***Structure***: Single byte read as a hexadecimal value<br>***Range:*** (0..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.<br>***Structure***: 32-bit word, containing dedicated values or bit fields.<br>***Range:*** (0..0xFFFFFFFF) |

## 4.3 FW_VERSION_INFO

At start-up, MGC3130 sends Fw_Version_Info message to the host interface to show that the chip is alive and ready for operation. Fw_Version_Info can also be requested using Request_Message (0x06).

**Note:** The payload elements HWRev, ParameterStartAddr and LibraryLoaderVersion are only valid after MGC3130 start-up.

Direction: MGC3130 to Host.

**TABLE 4-5: MESSAGE OVERVIEW**

| Header | | | | Payload | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Msg. Size | Flags | Seq. | ID | FwValid | HwRev | ParameterStartAddr | LibraryLoaderVersion | FwStartAddr | FwVersion |
| *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *2 Bytes* | *1 Byte* | *3 Bytes* | *1 Byte* | *120 Bytes* |
| 132 | n.a. | n.a. | 0x83 | see description below | | | | | |

**TABLE 4-6: PAYLOAD ELEMENTS**

| Element | Element Size (in bytes) | Description |
|---|---|---|
| FwValid | 1 | Status of GestIC® Library.<br>***Structure***: Single byte containing dedicated values (see list below)<br>***Possible values***:<br>0 Empty No valid GestIC® Library could be located<br>10 InvalidFW An invalid GestIC® Library was stored, or the last update failed<br>170 ValidFW A valid GestIC® Library is available |
| HwRev | 2 | Hardware revision information<br>***Structure***: Vector of 2 bytes interpreted as decimal values in format xx.xx<br>***Range***: (0..255, 0..255) |
| ParameterStartAddr | 1 | Parameter start address as supported by the Image<br>address = 128 * value of ParameterStartAddr<br>***Structure***: 1 byte interpreted as hex value<br>***Range***: (0..0xFF) |
| LibraryLoaderVersion | 3 | GestIC® Library loader version information<br>***Structure***: Vector of 3 bytes interpreted as decimal values in format xx.xx.xx<br>***Range***: (0..255, 0..255,0..255) |
| FwStartAddr | 1 | Start address of GestIC® Library as supported by the Bootloader, start address = 128 * value of FwStartAddr<br>***Structure***: 1 byte interpreted as hex value<br>***Range***: (0..0xFF) |
| FwVersion | 120 | Version information of GestIC® Library if valid (FwValid is not 0x00). The version string is interpreted as ASCII characters. It is a semicolon-separated string, always starting with the Version Number itself, followed by different tags.<br>**Supported Tags**:<br>p Platform (e.g. HillstarVxx)<br>DSP Colibri Suite Version (e.g. ID45r -1167)<br>s Reserved<br>c Reserved<br>t Build time (e.g. 2013/04/24 14:24:50)<br>***Structure***: Vector of 120 bytes interpreted as string (ASCII characters)<br>***Range***: (0..255, 0..255, 0..255, ...) |

## 4.4 SET_RUNTIME_PARAMETER

This message is used to set runtime 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 MGC3130.

**TABLE 4-7: 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* |
| 16 | n.a. | n.a. | 0xA2 | see description below | | | |

**TABLE 4-8: PAYLOAD ELEMENTS**

| Element | Element Size (in bytes) | Description |
|---|---|---|
| RuntimeParameterID | 2 | ID of runtime parameter. Please refer to **Section 4.4.1 "Trigger"** through **Section 4.4.12 "Data Output Lock Mask"**.<br>***Structure*:** 16-bit word interpreted as hex value<br>***Range:*** (0x0000..0xFFFF) |
| Reserved | 2 | write as '0' |
| Argument0 | 4 | Argument values, depending on runtime parameter ID. If not used, Argument0 should be provided as '0'.<br>***Structure***: 32-bit word: Argument0<br>***Range***: depends on runtime parameter |
| Argument1 | 4 | Argument values, depending on runtime parameter ID. If not used, Argument1 should be provided as '0'.<br>***Structure***: 32-bit word: Argument1.<br>***Range***: depends on runtime parameter. |

### 4.4.1 Trigger

This parameter forces a trigger defined in Argument0.

| | | |
|---|---|---|
| RuntimeParameterID | 0x1000 Trigger | Parameter forces a trigger. |
| Argument0 | 0: Force recalibration<br>1: Activate Processing mode<br>2: Enter Deep Sleep mode<br>***Range***: (0...2) | |
| Argument1 | Not used | |

### 4.4.2 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 | 0: Store RTPs for AFE<br>1: Store RTPs for DSP<br>2: Store RTPs for System<br>***Range***: (0...2) | |
| Argument1 | Not used | |

**Make Persistent Category: AFE**

### 4.4.3 Signal Matching

Signal matching parameters are used to adjust the Rx signal level at the sampling point.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x50 | afeRxAtt_S | Signal matching parameter for South electrode |
| | 0x51 | afeRxAtt_W | Signal matching parameter for West electrode |
| | 0x52 | afeRxAtt_N | Signal matching parameter for North electrode |
| | 0x53 | afeRxAtt_E | Signal matching parameter for East electrode |
| | 0x54 | afeRxAtt_C | Signal matching factor for Center electrode |
| Argument0 | Contains the value<br>***Range***: (0...255) | | |
| Argument1 | Not used | | |

### 4.4.4 Electrode Mapping

The physical channel number assigned to the electrodes. These parameters represent the physical connection of the electrodes to MGC3130 Rx channels. For the correct function, the mapping has to be looked up in the circuitry design.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x65 | Channelmapping_S | Physical channel assigned to the South Electrode |
| | 0x66 | Channelmapping_W | Physical channel assigned to the West Electrode |
| | 0x67 | Channelmapping_N | Physical channel assigned to the North Electrode |
| | 0x68 | Channelmapping_E | Physical channel assigned to the East Electrode |
| | 0x69 | Channelmapping_C | Physical channel assigned to the Center Electrode |
| Argument0 | Contains the number of physical receive channels (Rx0, Rx1, Rx2, Rx3, Rx4)<br>***Range***: (0...4) | | |
| Argument1 | Not used. | | |

**Make Persistent Category: DSP**

### 4.4.5 Transmit Frequency Selection

Sets the amount of used transmitter frequencies and the order in which they are tested for the frequency hopping.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x82 | TransFreqSelect | Parameter to set the used frequencies |
| Argument0 | Amount of used Tx frequencies | | |
| Argument1 | This determines in what order the transmitter frequencies are tested. The indexes numbered 0 to 4 represent respective transmitter frequencies. These indexes have to be provided in half bytes. | | |
| 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.<br>e.g. Index - Frequency Mapping for Hillstar<br>Index 0 - Transmitter Frequency: 115 kHz<br>Index 1 - Transmitter Frequency: 103 kHz<br>Index 2 - Transmitter Frequency: 88 kHz<br>Index 3 - Transmitter Frequency: 67 kHz<br>Index 4 - Transmitter Frequency: 44 kHz | | |

### 4.4.6 Touch Detection

This parameter enables/disables Touch Detection.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x97 | dspTouchConfig | Parameter to enable/disable Touch Detection |
| Argument0 | Set Argument0 to '0x08' to enable and set Argument0 to '0x00' to disable Touch Detection<br>**Note:** If Argument1 is not set correctly the system will show malfunctions. | | |
| Argument1 | 0x08 | | |

### 4.4.7 Approach Detection

This parameter enables/disables Approach Detection mode.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x97 | dspAutoWakeupMode | Parameter to enable/disable Approach Detection Mode |
| Argument0 | Set Argument0 to 0x01 to enable and set Argument0 to 0x00 to disable Approach Detection<br>**Note:** If Argument1 is not set correctly the system will show malfunctions. | | |
| Argument1 | 0x01 | | |

### 4.4.8 AirWheel

This parameter enables/disables AirWheel.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x90 | dspAirWheelConig | Parameter to enable/disable AirWheel |
| Argument0 | Set Argument0 to '0x20' to enable and set Argument0 to '0x00' to disable AirWheel<br>**Note:** If Argument1 is not set correctly the system will show malfunctions. | | |
| Argument1 | 0x20 | | |

**Make Persistent Category: System**

### 4.4.9 Gesture Processing (HMM)

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

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x85 | dspGestureMask | Parameter to enable/disable gestures |
| Argument0 | Bit 0: Garbage model<br>Bit 1: Flick West to East<br>Bit 2: Flick East to West<br>Bit 3: Flick South to North<br>Bit 4: Flick North to South<br>Bit 5: Circle clockwise<br>Bit 6: Circle counter-clockwise | | |
| Argument1 | Not used | | |

### 4.4.10 Calibration Operation Mode Flags

This parameter enables/disables the auto-calibration feature.

| | | | |
|---|---|---|---|
| RuntimeParameterID | 0x80 | dspCalOpModeFlags | Parameter to enable/disable auto-calibration |
| Argument0 | 0x00: Disable auto-calibration<br>0x3F: Enable auto-calibration | | |
| Argument1 | 0x00: Disable auto-calibration<br>0x3F: Enable auto-calibration | | |

### 4.4.11 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)`.

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

Please mind that 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` | Bit 0: DSP Status |
| | Bit 1: Gesture Data |
| | Bit 2: Touch Data |
| | Bit 3: AirWheel Data |
| | Bit 4: Position Data |
| | Bit 5: Noise Power |
| | Bits 6...10: These bits are reserved and must be set to '`0`'. |
| | Bit 11: Uncalibrated Signal (CIC) Data. |
| | Bit 12: Signal Deviation (SD) Data. |
| | 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 are kept unchanged. |

### 4.4.12 Data Output Lock 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)` no matter whether there is new data or not (payload element is "locked").

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 variable).

| | | | |
|---|---|---|---|
| `RuntimeParameterID` | `0xA1` | `DataOutputLockMask` | Parameter determining the data output. |

| | |
|---|---|
| `Argument0` | Bit 0: DSP Status<br>Bit 1: Gesture Data<br>Bit 2: Touch Data<br>Bit 3: AirWheel Data<br>Bit 4: Position Data<br>Bit 5: Noise Power<br>Bits 6...10: These bits are reserved and must be set to '`0`'.<br>Bit 11: Uncalibrated Signal (CIC) Data.<br>Bit 12: Signal Deviation (SD) Data.<br>Bit 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 are kept unchanged. |

## 4.5 SENSOR_DATA_OUTPUT

This message contains the sensor data output of the MGC3130. The content of the message can be configured via bit mask (refer to `DataOutputEnableMask` and `DataOutputLockMask` in **Section 4.4 "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: MGC3130 to Host

**TABLE 4-9: 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 4-10: PAYLOAD ELEMENTS**

| Element | Element size (in bytes) | Description |
|---|---|---|
| `DataOutputConfig-Mask` | 2 | Bit mask indicating which data is part of the message.<br>The following bits are used:<br>Bit 0: `DSPInfo` field.<br>Bit 1: `GestureInfo` field.<br>Bit 2: `TouchInfo` field.<br>Bit 3: `AirWheelInfo` field.<br>Bit 4: `xyzPosition` field.<br>Bit 5: `NoisePower` field.<br>Bit 6: This bit is reserved.<br>Bit 7: This bit is reserved.<br>Bit 8...10: `ElectrodeConfiguration`<br>`000`: `ChCnt` = 4, four electrode configuration w/o Center electrode<br>`001`: `ChCnt` = 5, five electrode configuration with Center electrode<br>Bit 11: CICData field with `chCnt` entries.<br>Bit 12: SDData field with `chCnt` entries.<br>Bit 13...15: These bits are reserved.<br>***Structure***: 16-bit word read as a bit mask.<br>***Range***: (`0000000000000000b..1111111111111111b`) |
| `TimeStamp` | 1 | 8-Bit Counter of 200 Hz (Sample Interval)<br>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.<br>***Structure***: 8-bit word read as decimal value.<br>***Range***: (`0..255`) |
| `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`'.<br>The following bits are used:<br>Bit 0: `PositionValid`, if set indicates that the position in the `xyzPosition` field is valid.<br>Bit 1: `AirWheelValid`, if set indicates that the `AirWheel` is active and the data in the `AirWheelInfo` field is valid.<br>Bit 2: `RawDataValid`, if set indicates that the data of the `CICData` and `SDData` fields are valid. Otherwise those fields must be ignored.<br>Bit 3: `NoisePowerValid`, if set indicates that the `NoisePower` field is valid.<br>Bit 4: `EnvironmentalNoise`, if set indicates that environmental noise has been detected.<br>Bit 5: Clipping, if set indicates that the ADCs are clipping.<br>Bit 6: This bit is reserved.<br>Bit 7: `DSPRunning`, if set indicates that the system is currently running. If not set, the system is about to go to sleep.<br>***Structure***: 8-bit word read as a bit mask.<br>***Range***: (`00000000b..1111111b`) |
| `DSPInfo` | 2 | This element consists of two bytes. The first byte contains information about calibration events. The second byte indicates the Tx frequency currently used.<br>Bit 0: This bit is reserved.<br>Bit 1: `CalibrationInfo`: Forced calibration (by Host)<br>Bit 2: `CalibrationInfo`: Start-up calibration<br>Bit 3: `CalibrationInfo`: Gesture triggered<br>Bit 4: `CalibrationInfo`: Negative value<br>Bit 5: `CalibrationInfo`: Idle calibration<br>Bit 6: `CalibrationInfo`: Invalid value calibration<br>Bit 7: This bit is reserved.<br>Bits 8…15: Tx Frequency in kHz gesture as decimal value (`44..115`)<br>***Structure***: 2 bytes, first byte read as a bit mask second byte as decimal.<br>***Range***: (`00000000b..11111111b`; `44..115`) |

**TABLE 4-10: PAYLOAD ELEMENTS (CONTINUED)**

| Element | Element size (in bytes) | Description |
|---|---|---|
| GestureInfo | 4 | This field contains the 32-bit gesture information word.<br>***Recognized Gestures***:<br>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:<br>Bits 7...0: Recognized gesture as decimal number<br>0: No gesture<br>1: Garbage model<br>2: Flick West to East<br>3: Flick East to West<br>4: Flick South to North<br>5: Flick North to South<br>6: Circle clockwise (only active if AirWheel disabled)<br>7: Circle counter-clockwise (only active if AirWheel disabled)<br>Bits 8...14: These bits must not be interpreted.<br>Bits 15…12: Gesture Class read as a decimal number<br>0: Garbage model<br>1: Flick gesture<br>2: Circular gesture<br>Bit 16: Edge flick – is 1 if flick gesture is classified as edge flick<br>Bits 17...31: These bits are reserved.<br>***Structure***: 32-bit word read as a bit mask<br>***Range***: (00000000000000000000000000000000b..11111111111111111111111111111111b) |
| TouchInfo | 4 | Contains touch information<br>The following bits are used to indicate a touch event on the respective electrodes:<br>Bit 0: Touch South electrode<br>Bit 1: Touch West electrode<br>Bit 2: Touch North electrode<br>Bit 3: Touch East electrode<br>Bit 4: Touch Center electrode<br>Bit 5: Tap South electrode<br>Bit 6: Tap West electrode<br>Bit 7: Tap North electrode<br>Bit 8: Tap East electrode<br>Bit 9: Tap Center electrode<br>Bit 10: Double Tap South electrode<br>Bit 11: Double Tap West electrode<br>Bit 12: Double Tap North electrode<br>Bit 13: Double Tap East electrode<br>Bit 14: Double Tap Center electrode<br>Bits 15...31: These bits must not be interpreted.<br>***Structure***: 32-bit word read as a bit mask<br>***Range***: (00000000000000000000000000000000b..11111111111111111111111111111111b) |
| AirWheelInfo | 2 | The first byte contains a counter which indicates how far the AirWheel rotation has progressed. Incrementing values indicate a clockwise rotation. Decrementing values indicate counter clockwise rotation. An increment of 32 approximates one full rotation. AirWheelInfo is only valid if the AirWheelValid bit in the element SystemInfo is '1'.<br>The second byte is reserved.<br>***Structure***: Vector of two 8-bit words read as a decimal value<br>***Range***: (0..255, reserved) |
| xyzPosition | 6 | This element contains x, y and z position data. Two bytes are used for each of the positions x, y and z.<br>Bytes 1 and 2: x position<br>Bytes 3 and 4: y position<br>Bytes 5 and 6: z position<br>The position information is only valid if the PositionValid bit in the element SystemInfo is '1'.<br>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.<br>***Structure***: Vector of three16-bit words read as a decimal value for each position x, y, z<br>***Range***: (0..65535) for each position x, y, z |
| NoisePower | 4 | Noise Power of the GestIC® system.<br>NoisePower is only valid if the NoisePowerValid bit in the element SystemInfo is '1'.<br>***Structure***: 32-bit word read as a float value<br>***Range***: (0..3.402823e+38) |
| CICData | 4xChCnt | Uncalibrated Sensor Data (CIC Data)<br>Element size depends on ChCnt indicated in payload element DataOutputConfigMask bits 8...10.<br>CICData are only valid if the RawDataValid bit in the element SystemInfo is '1'.<br>***Structure***: Vector of four, respectively five, 32-bit words interpreted as float values in format xxxx.xxxx.xxxx.xxxx.xxxx (South.West.North.East.Center)<br>***Range***: (-3.402823e+38..3.402823e+38) for each channel |

**TABLE 4-10: PAYLOAD ELEMENTS (CONTINUED)**

| Element | Element size (in bytes) | Description |
|---|---|---|
| SDData | 4xChCnt | Signal Deviation (SD)<br>Element size depends on ChCnt indicated in payload element DataOutputConfigMask bits 8...10.<br>SDData are only valid if the RawDataValid bit in the element SystemInfo is '1'.<br>***Structure***: Vector of four, respectively five, 32-bit words interpreted as float values in format xxxx.xxxx.xxxx.xxxx.xxxx (South.West.North.East.Center)<br>***Range***: (-3.402823e+38...3.402823e+38) for each channel |
| Reserved | | Reserved: Additional payload elements can be added in the future or for internal purposes. |

# Chapter 5. Messages for GestIC Library Update

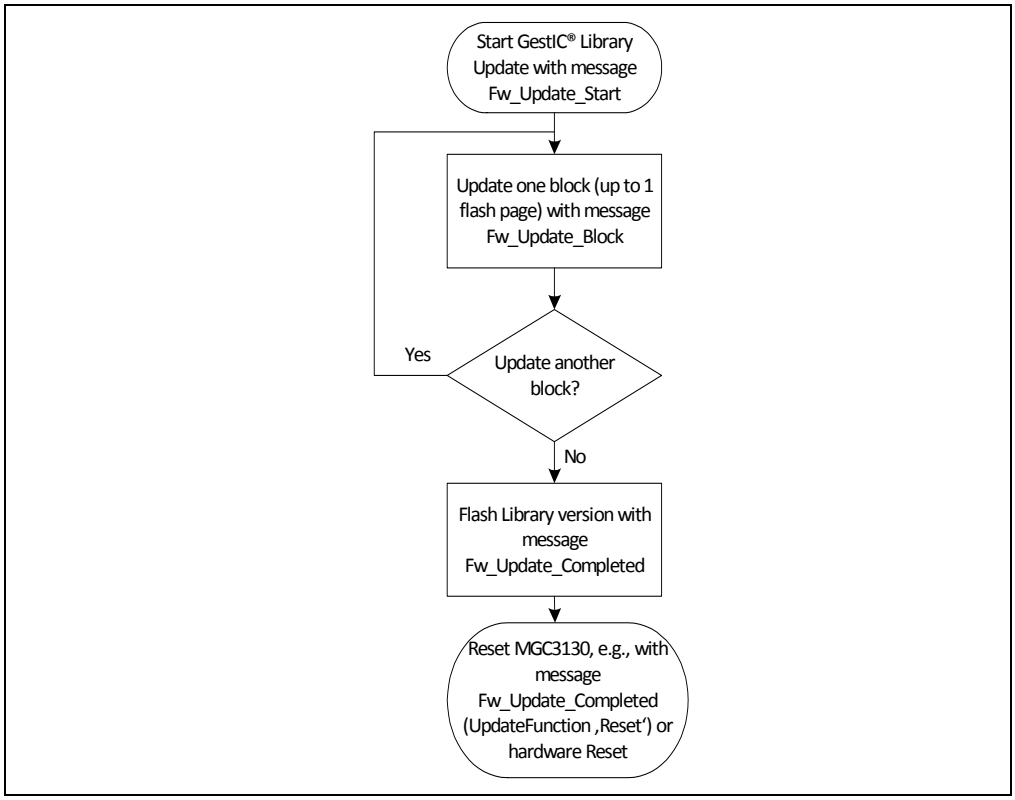
## 5.1 LIBRARY LOADER UPDATE PROCEDURE

The general library update process is shown in Figure 5-1. Please note that only libraries provided by Microchip Technology can be updated on the MGC3130. Furthermore, an Application Note which describes the library update process in detail can be delivered by Microchip by request only.

For the library update process, three different messages are required:

- `Fw_Update_Start` (Message ID - `0x80`)
- `Fw_Update_Block` (Message ID - `0x81`)
- `Fw_Update_Completed` (Message ID - `0x82`)

**FIGURE 5-1: LIBRARY UPDATE FLOWCHART**



## 5.2 FW_UPDATE_START

This message starts the update session of the MGC3130 device.

Direction: Host to MGC3130

**TABLE 5-1: MESSAGE OVERVIEW**

| Header | | | | Payload | | | | |
|---|---|---|---|---|---|---|---|---|
| Msg. Size | Flags | Seq. | ID | Crc | SessionID | IV | UpdateFunction | Reserved |
| *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *4 Bytes* | *4 Bytes* | *14 Bytes* | *1 Byte* | *1 Byte* |
| 28 | n.a. | n.a. | 0x80 | see description below | | | | |

**TABLE 5-2: PAYLOAD ELEMENTS**

| Field | Size (in bytes) | Description |
|---|---|---|
| Crc | 4 | A Crc32, calculated across the following fields (Address, Length and Payload)<br>***Structure***: 32-bit word<br>***Range***: (0x00000000..0xffffffff) |
| SessionId | 4 | The SessionID is a random number generated by the Host. It has to be resent in the Fw_Update_Completed message or else the session will be invalid. 0x00000000 is an invalid SessionID and is used to force the device in a wait loop. In this case, the remaining information in this message is discarded.<br>***Structure***: 32-bit word<br>***Range***: (0x00000000..0xffffffff) |
| IV | 14 | 14-byte value which is used to encrypt the data.<br>***Structure***: Vector of 14 bytes<br>***Range***: (0..255, 0..255, 0..255, ...) |
| UpdateFunction | 1 | The UpdateFunction sets the mode of the whole update session:<br>- If the Session mode is set ProgramFlash, the Payloads of the following Fw_Update_Block messages are written to Flash.<br>- If the Session mode is set VerifyOnly, the code is only verified (comparison between Flash content and decrypted payload of Fw_Update_Block messages), but not written to Flash. If a mismatch between decrypted payload and Flash is found, a System_Status message with an Error 8 (ContentMismatch) is returned<br>**Note:** The following Fw_Update_Block messages also contain an UpdateFunction field. That field defines the mode for the single Update Blocks.<br>However:<br>- if the mode of the session is set to ProgramFlash via Fw_Update_Start, the UpdateFunction of the single Fw_Update_Blocks can be set to ProgramFlash or to VerifyOnly.<br>- if the mode of the session is set to VerifyOnly via Fw_Update_Start, the UpdateFunction of the single Fw_Update_Blocks can only be set to VerifyOnly.<br>***Structure***: Single byte containing dedicated values (see list below)<br>***Possible values***:<br>0 Program Flash<br>1 VerifyOnly |
| Reserved | 1 | Reserved |

## 5.3 FW_UPDATE_BLOCK

This message updates one block of the Flash. The size of one block can be up to 128 bytes.

Direction: Host to MGC3130

**TABLE 5-3: MESSAGE OVERVIEW**

| Header | | | | Payload | | | | |
|---|---|---|---|---|---|---|---|---|
| Msg. Size | Flags | Seq. | ID | Crc | Address | Length | UpdateFunction | Payload |
| *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *4 Bytes* | *2 Bytes* | *1Byte* | *1 Byte* | *128 Bytes* |
| 140 | n.a. | n.a. | 0x81 | see description below | | | | |

**TABLE 5-4: PAYLOAD ELEMENTS**

| Field | Size (in bytes) | Description |
|---|---|---|
| Crc | 4 | Crc32 value, calculated across the rest of the message<br>***Structure***: 32-bit word<br>***Range***: (0x00000000..0xffffffff) |
| Address | 2 | The Flash address of the block which will be programmed/verified. If the block is smaller than 128 bytes, it has to be aligned at the end of each page. So, if the next update block is a full 128-byte block, it can be Flash-page aligned again.<br>**Note:** The lower 4 KB are reserved for the Library Loader and cannot be updated. If a value lower than the 4 KB is used, a System_Status message with the Error 5 (InvalidAddress) is returned.<br>***Structure***: 16-bit word<br>***Range***: (0x1000..0x7fff) |
| Length | 1 | The length of the content of the block which will be updated:<br>***Structure***: Single byte<br>***Range***: (0..128) |
| UpdateFunction | 1 | The UpdateFunction sets the mode for this single Update Block.<br>- If the mode is set ProgramFlash, the decrypted Payload is written to Flash.<br>- If the Session mode is set VerifyOnly, the code is only verified (comparison between Flash content and decrypted payload, but not written to Flash. If a mismatch between decrypted payload and Flash is found, a System_Status message with Error 8 (ContentMismatch) is returned.<br>**Note:** If the mode of the whole session was set to VerifyOnly in the Fw_Update_Start message, only VerifyOnly can be set in the Fw_Update_Block; otherwise, a System_Status message with Error 6 (InvalidFunction) is returned.<br>***Structure***: Single byte containing dedicated values (see list below)<br>***Possible values***:<br>0 ProgramFlash<br>1 VerifyOnly |
| Payload | 128 | The Payload contains the encrypted content of the block which will be updated.<br>**Note:** Its length is always 128. If the length of the content is smaller than 128, it will be filled with zeros. The Crc is then calculated over the entire 128-byte Payload.<br>***Structure***: Vector of 120 bytes interpreted as String (ASCII characters)<br>***Range***: (0..255, 0..255, 0..255, . ..) |

## 5.4 FW_UPDATE_COMPLETED

This message finalizes the update session of the MGC3130.

Direction: Host to MGC3130

**TABLE 5-5: MESSAGE OVERVIEW**

| Header | | | | Payload | | | | |
|---|---|---|---|---|---|---|---|---|
| Msg. Size | Flags | Seq. | ID | Crc | SessionID | UpdateFunction | FwVersion | Reserved |
| *1 Byte* | *1 Byte* | *1 Byte* | *1 Byte* | *4 Bytes* | *4 Bytes* | *1 Byte* | *120 Bytes* | *3 Bytes* |
| 136 | n.a. | n.a. | 0x82 | see description below | | | | |

**TABLE 5-6: PAYLOAD ELEMENTS**

| Field | Size (in bytes) | Description |
|---|---|---|
| Crc | 4 | Crc32 value, calculated across the rest of the message<br>***Structure***: 32-bit word<br>***Range***: (0x00000000..0xffffffff) |
| SessionID | 4 | The SessionID is the same random number as used for the Fw_Update_Start. 0x00000000 is an invalid SessionID and forces the device into a restart. In this case, the remaining information in this message is discarded.<br>**Structure**: 32-bit word<br>***Range***: (0x00000000..0xffffffff) |
| UpdateFunction | 1 | The UpdateFunction defines how the update session is finalized.<br>- If the session was started as ProgramFlash session, it has to be finalized with the ProgramFlash session. If not, the library version is not stored and the library is not valid. If ProgramFlash is used in a VerifyOnly session, a System_Status message with Error 6 (InvalidFunction) is returned.<br>- If Restart is used, the device will restart. FwVersion and SessionID are included in Crc calculation, but content is ignored.<br>***Structure***: Single byte containing dedicated values (see list below)<br>***Possible values***:<br>0 ProgramFlash Program Flash<br>1<br>2 Restart |
| FwVersion | 120 | It contains the library version. Only libraries with IDs other than 0 are valid.<br>***Structure***: Vector of 120 bytes interpreted as String (ASCII characters)<br>***Range***: (0..255, 0..255, 0..255, . ..) |
| Reserved | 3 | Reserved |

# Appendix A. Glossary

**TABLE A-1: GLOSSARY**

| Term | Definition |
|---|---|
| AFE | Analog front end |
| Application Host | PC or embedded controller which controls the MGC3130 |
| Aurea | MGC3130 PC control software with graphical user interface |
| Colibri Suite | Embedded DSP suite within the GestIC® Library |
| Deep Sleep | MGC3130 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 MGC3130 features and is delivered as a binary file preprogrammed on the MGC3130 |
| 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 |
| MGC3130 | Single-Zone 3D Gesture Sensing Controller |
| Position Tracking | GestIC® technology feature |
| Sabrewing | MGC3130 evaluation board |
| Self Wake-up | MGC3130 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<br>(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 MGC3130 with approach detection |

# 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

**Cleveland**
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

**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

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110

**Canada - Toronto**
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Chongqing**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**China - Hangzhou**
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

**China - Hong Kong SAR**
Tel: 852-2943-5100
Fax: 852-2401-3431

**China - Nanjing**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-3019-1500

**Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**Japan - Tokyo**
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

**Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

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

**Taiwan - Taipei**
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

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

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

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

**Germany - Dusseldorf**
Tel: 49-2129-3766400

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

**Germany - Pforzheim**
Tel: 49-7231-424750

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

**Italy - Venice**
Tel: 39-049-7625286

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

**Poland - Warsaw**
Tel: 48-22-3325737

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

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

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

10/28/13