MAGENSA™
A MAGTEK COMPANY

DynaWave

Firmware Update through UART Port on mDynamo
Programmer's Guide

June 2021

Document Number:
D998200482-10

REGISTERED TO ISO 9001:2015

Magensa, LLC I 1710 Apollo Court I Seal Beach, CA 90740 I Phone: (562) 546-6400 I Technical Support: (888) 624-8350
www.magtek.com

Table 0-1 - Revisions

Rev Number	Date	Notes
10	06/14/2021	Initial Release

LIMITATION ON LIABILITY

EXCEPT AS PROVIDED IN THE SECTIONS RELATING TO MAGTEK'S LIMITED WARRANTY, MAGTEK'S LIABILITY UNDER THIS AGREEMENT IS LIMITED TO THE CONTRACT PRICE OF THIS PRODUCT.

MAGTEK MAKES NO OTHER WARRANTIES WITH RESPECT TO THE PRODUCT, EXPRESSED OR IMPLIED, EXCEPT AS MAY BE STATED IN THIS AGREEMENT, AND MAGTEK DISCLAIMS ANY IMPLIED WARRANTY, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

MAGTEK SHALL NOT BE LIABLE FOR CONTINGENT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES TO PERSONS OR PROPERTY. MAGTEK FURTHER LIMITS ITS LIABILITY OF ANY KIND WITH RESPECT TO THE PRODUCT, INCLUDING NEGLIGENCE ON ITS PART, TO THE CONTRACT PRICE FOR THE GOODS.

MAGTEK'S SOLE LIABILITY AND BUYER'S EXCLUSIVE REMEDIES ARE STATED IN THIS SECTION AND IN THE SECTION RELATING TO MAGTEK'S LIMITED WARRANTY.

Table of Contents

1 The Purpose of This Document

This is a programmer's guide to update the firmware of DynaWave through mDynamo. It contains instructions on how to update the DynaWave firmware in the case where DynaWave is only accessible through the UART port on the mDynamo.

2 Hardware Setup

Host should connect to mDynamo through USB, while DynaWave should be connected to the UART port on the mDynamo.

3 Baud Rate Configuration

To achieve maximum through put and least time for firmware update process, the UART baud rate on mDynamo should be set to 115200. For instructions on how to set baud rate, please refer to MagTek documentation "D998200351-11 -Update and Test Baud Rate - Remote Services v2.0".

4 Firmware Update Procedures

Following steps should be implemented in sequence, as the following steps depend on the previous step(s).

4.1 Step 1: Get firmware KSN from DynaWave.

There are two keys on DynaWave, one for transaction, and the other for EMV Config and Firmware Update. For firmware update, the corresponding firmware key needs to be used, thus the firmware KSN. To get the firmware KSN, following 2 steps can be followed:

- Step 1a: Use command 2100 to retrieve DynaWave UIK.

  Following is one example of the response:
  "001F42353138364443303531313231414100**9010010B5186DC000000**01B5EEE6150"

- Step 1b: Construct the KSN by taking sub string of UIK (37~55) and append by "1".

  Following is the constructed firmware KSN:
  "9010010B5186DC000001"

Please note that the counter (last portion of the KSN) is fixed to be 1 because the firmware KSN is fixed and never increment.
If the DynaWave is plugged into the UART port of mDynamo, and the host is connected to mDynamo directly, command 2100 need to be formatted in SLIP format before it's sent to mDynamo from the host. Please refer to Command Format and Response Format sections below on detail of SLIP format.

4.2 Step 2: Find the Key ID for firmware update.

Remote Services will be used to generate the firmware update commands. Key ID is one of the required inputs for remote service call. To find the Key ID, following steps can be followed:

- Step 2a: Derive KSI from the firmware KSN (generated from Step 1) by taking sub string of the KSN, (1~7). i.e. "9010010".
- Step 2b: Call Remote Services (operation SCRAv2.svc::GetKeyList) to retrieve the assigned key list.
- Step 2c: Find the key from the key list with the matching KSI identified in Step 2a.

- Step 2d: The key ID of the key found will be used to generate firmware update commands. Please note, if there is no key found in Step 2c, it means you don't have permission to that key. Please contact customer service regarding the key permission.

4.3 Step 3: Generate firmware update commands through Remote Services

Operation SCRAv2.svc::GetFirmwareCommands should be used with the following parameters:

- KSN: the KSN found in Step 1.
- KeyID: the key ID found in Step 2.
- DeviceType: "mDynamo".
- SerialNumber: leave it blank. This is reserved for future use.
- Firmware: the bye array of the actual firmware.

The response to the service call contains 3 parts: Preload Commands, Postload Commands, and the Firmware Commands. Preload and Postload Commands contain MACROs that the host application needs to interpret accordingly. MACRO may be following by data or command, in which case they are separated by space. Here is the list of macros:

- NOMAC: the command followed will be used as is and sent to device with SLIP format.
- WAIT: the number followed will be used as milliseconds. Host application needs to wait before sending the next command.
- DETECTDEVICE: refresh the device list.

Following is an example of the Preload Commands list:

NOMAC 6800
WAIT 10000
DETECTDEVICE
NOMAC 6000

Following is an example of the Postload Commands list:

NOMAC 6200
NOMAC 6600
NOMAC 6700
WAIT 10000

The third part of the response is the actual Firmware Commands. They will be sent to the device with SLIP format after the Preload Commands are processed.

4.4 Step 4: Process Preload Commands

The Preload Commands should be parsed and processed by the Host application according to the MACRO definition in Step 3. Commands sent to the device will need to be formatted in SLIP format described in Command Format section. Below are details of the device related commands:

- 6800: Run bootloader ODM
- 6000: Initialize Main ODM

4.5 Step 5: Process Firmware Commands

The Firmware Commands will be sent to device one by one in SLIP format. As it is detailed in Response Format section, both the regular response and the extended response need to be received for each command before the Host sends the next command to the device. Because the daisy-chain configuration, the command will be processed by device slower than usual.

4.6 Step 6: Process Poastload Commands

The Postload Commands should be parsed and processed by the Host application according to the MACRO definition in Step 3. Commands sent to the device will need to be in SLIP format as described in Command Format section. Below are details of the device related commands:

- 6200: Close Main ODM
- 6600: Verify Main ODM
- 6700: Finalize Main ODM

Please note that commands 6600 & 6700 will take longer than regular commands. So the Host application should expect longer than usual time for the extended response to come back to the host.

5 Command Format

To send command to DynaWave from mDynamo through UART port, each command needs to be constructed in SLIP format. The constructed command will be sent to mDynamo in extended format using MTSCR::sendExtendedCommand.
For instance, to issue command 6800 (the first command of the Preload Commands), the constructed command will be "0400000800C00500026800C0". Here is the explanation of each part:

- 0400 -> Auxiliary UART Transmit Data command
- 0008 -> Data length
- 00 -> Port Identifier
- C0 -> Beginning of SLIP format
- 05 -> Message Type: Command Request
- 0002 -> Command Length (big endian)
- 6800 -> Original command
- C0 -> Closing SLIP Format

Please note that Host application needs to take into account SLIP escape sequences that deal with occurrences of C0 inside the SLIP data frame:

- If outbound data contains the byte value **C0**, software should encode it into SLIP as **DB DC**; if inbound SLIP data contains the byte sequence **DB DC**, software should decode it to **C0**.
- If outbound data contains the byte value **DB**, software should encode it into SLIP as **DB DD**; if inbound SLIP data contains the byte sequence **DB DD**, software should decode it to **DB**.

Here are sample codes (C#) for constructing SLIP format command:

```
private string GeneratePassthroughCmd(string cmd)
{
    string cmdSLIP = applySLIPProtocol(cmd);

    string data = "00C005" + (cmd.Length / 2).ToString("X4") + cmdSLIP + "C0";

    return "0400" + (data.Length / 2).ToString("X4") + data;
}
```

```
private string applySLIPProtocol(string cmd)
{
    StringBuilder sb = new StringBuilder();
    for(int i=0; i<cmd.Length; i+=2)
    {
        string byteCode = cmd.Substring(i, 2).ToUpper();
        if (byteCode == "C0")
            sb.Append("DBDC");
        else if (byteCode == "DB")
            sb.Append("DBDD");
        else
            sb.Append(byteCode);
    }
    return sb.ToString();
}
```

Please note that the command length after "C005" is the original length before any SLIP protocol escaping.

For instance, firmware command

"613A15FB13221820454361193000014F046FB0020F2BC08BC18470BFEFFFFFEB500273D00264807F0D2FA0400012011306D1C08306D1C C0 1C6D1C401D"

will be transformed to

"0400004300C005003C 613A15FB13221820454361193000014F046FB0020F2BC08BC18470BFEFFFFFEB500273D00264807F0D2FA0400012011306D1C08306D1C DBDC 1C6D1C401D C0"

Please note that spaces are inserted for demonstration purpose only.

6 Response Format

Each extended command constructed in SLIP format will have two separate responses from mDynamo. First response is from MTSCRA::OnDeviceResponse event handler from the SDK. This response should be "0000" which indicates the command has been successfully processed my mDynamo.
The second response is from MTSCRA:: OnDeviceExtendedResponse event handler. This response must be received before the Host application sends the next command. This is to ensure all data packets are passed through from mDynamo to DynaWave successfully. If next extended command is sent to mDynamo before this response is received, the data packet might be dropped, causing firmware not complete and final verification to fail.
Here is an example of the extended response: "0400000800C00400020000C0." Below is the explanation of each part:

- 0400 -> Auxiliary UART Transmit Data command
- 0008 -> Data length
- 00 -> Port Identifier
- C0 -> Beginning of SLIP format
- 04 -> Message Type: Command Response
- 0002 -> Command Length (big endian)
- 0000 -> Original response (Success)
- C0 -> Closing SLIP Format

7 Programming Considerations

The Host application needs to consider the followings:

- For each extended command sent to mDynamo, the Host application needs to receive both regular response and the extended response. Both responses are sent to the Host by mDynamo at different time frame. If either one of the responses is not received, or either one of the responses is not a success response, the process should be terminated. First response will take a few milliseconds, while the second response might take 50ms to 100ms (a few might take even longer out of the total thousands of generated firmware commands).
- Host application will only send next command after receiving both responses mentioned above. This is to make sure each data packet is processed completely without data loss.
- The last two Postload Commands, 6600 & 6700 will take longer than usual. Several seconds is expected as normal.

Retrieving and Processing of the firmware update commands should be considered two separate processes for easier programming. During processing of the commands, either timer or event/delegate based programming model can be used to interact between host and the devices.

8 Implementation Observation

On Windows platform using Windows SDK to interact with the device, please take a note on the following observations:

- 100ms delay between setting each firmware commands will work for most of the commands. A small portion of the commands might need longer time for the extended response to come back. So instead of increasing the delay time for each command, you might want to have second timer (as a timeout timers) so extra wait time will be given for those needed. This improves overall process time. If you decrease the delay to 50ms, some more commands will need extra wait time, but the overall process time might be shorter.
- The timeout timer can be set to no less than 1 second. For the postload command, it will take longer, and they can be handled separately if desired.
- The firmware update might take between 10 to 20 minutes.
- If the data is sent to device with receiving the extended response from the previous command, the data might be lost.
- If command is not correctly escaped in SLIP format, it might be lost with no extended response coming back.