

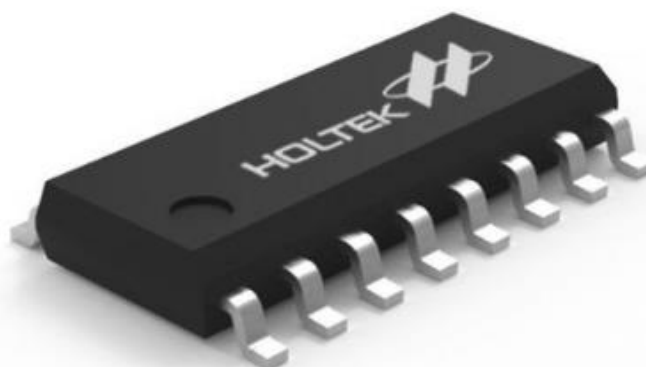
Interfacing FlashRunner with HOLTEK 8-bit General Purpose MCUs (HT66F00x1)

1. Introduction

The devices are **Flash Memory A/D type 8-bit high performance RISC architecture microcontrollers**. Offering users the convenience of Flash Memory multi-programming features, the devices also include a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of **Emulated EEPROM** memory for storage of non-volatile data such as serial numbers, calibration data, etc.

Analog feature includes a multi-channel 10-bit A/D converter. A single extremely flexible Timer/Event Counter provides timing, event counting and pulse wide capture functions. A Pulse Width Modulator provides an 8-bit PWM output. An internal Watchdog Timer coupled with excellent noise immunity and ESD protection ensures that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low speed oscillators are provided and the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize power consumption.



You can download the latest version of this document from this static link: [Interfacing FlashRunner with HOLTEK 8-bit General Purpose MCUs](#).

2. Contents

1.	Introduction	1
2.	Contents.....	2
3.	Tips and tricks to flash HT66F00x1 MCUs	3
4.	Option Bytes	6
	HT66F00x1	6
5.	Security Management	9
	HT66F00x1	9

3. Tips and tricks to flash HT66F00x1 MCUs

In this section, we want to explain which operations are performed in the target devices and how these operations work. This knowledge may help you to optimize even more your application.

FlashRunner uses **TYPE-23** custom protocol to access memories of these devices.

TYPE-23 is a *2-wired serial synchronous interface* that makes use of the following signals:

1. **IPCK** (Serial clock)
2. **IPCD A** (Tx and Rx data)

Let us now analyse the standard commands:

- **MASSERASE**

This command erases all the contents of selected memory.

Supported memories for this command are:

- Program ROM [F]
- Emulated EEPROM [E]
- Option ROM [O]
- Chip Erase [C]

Optionally, it is also possible to send additional parameters to this command to select only some blocks of memory to be erased. This can be useful to reduce the erase time and it could be needed if the user wants to preserve some data into the memory and partially reprogram it. See **Block Erase** chapter for more information.

One suggestion is to skip this operation in case the device is virgin (i.e. factory shipped). This can be done using a conditional script based on the result of the blankcheck operation:

```
#IFERR TPCMD BLANKCHECK F
#THEN TPCMD MASSERASE F
#THEN TPCMD BLANKCHECK F
```

Note: performing MASSERASE Option will not erase last 3 Options, that are written by default from Manufacturer. Blankcheck will also skip these specific locations.

In case certain areas cannot be erased (and that depends on target device), it is still possible to erase their content with the following command:

#TPCMD MASSERASE C

This command will fully erase **Program ROM**, **Emulated EEPROM** and **Option ROM** so once issued, it could then be followed by a **#TPCMD BLANKCHECK** command for all the target memories and no further masserases are necessary.



- **BLANKCHECK**

This command checks that all the bits of the selected memory are set to 0 (erased state).

Supported memories for this command are:

- Program ROM [F]
- Emulated EEPROM [E]
- Option ROM [O]

This operation is executed internally by the device and it is typically extremely fast.

Optionally, it is also possible to send two additional parameters to this command: the address from where to start checking and the number of bytes to check.

The duration of this operation depends on the size of the memory to check and communication speed doesn't speed up the process.

- **PROGRAM**

This command takes the Customer's data from the FRB file and programs them into selected memory.

Supported memories for this command are:

- Program ROM [F]
- Emulated EEPROM [E]
- [Option ROM](#) [O]

The duration of this command depends on many factors which are all important:

- The target device characteristics, in other words, how much time is needed to write data into the memory.
- The target device frequency because it determines how fast the device is able to process incoming data.
- Protocol frequency because it sets the bitrate of the communication between FlashRunner and the target device.

- **VERIFY READOUT**

This command checks that data contained in the memory of the device corresponds to FRB data.

Supported memories for this command are:

- Program ROM [F]
- Emulated EEPROM [E]

This command works exactly like the program command with the only exception that the device reads (instead of writing) data from the memory and compares that with the data received from FlashRunner.

- **TRIM**

This command performs the Trim operation, useful for correctly calibrating the chip; the calibration values are inserted in 3 specific positions in the Option ROM. It is possible to select if execute the operation at 3000mV or 5000mV. (Operation is enabled by default)

✓ TPCMD TRIM

3000 ▾

#TPCMD TRIM <3000|5000>


4. Option Bytes

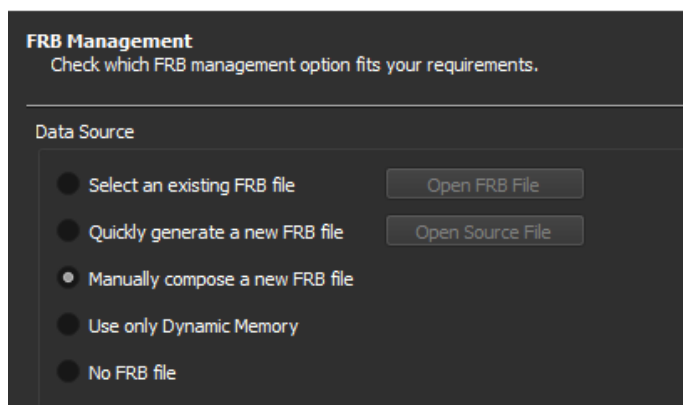
HT66F0021 have a set of non-volatile registers where boot relevant data are taken from and loaded into corresponding volatile registers once the reset is released. They are known as **Option Bytes**. They are **calculated automatically from Flashrunner**.

HT66F00x1

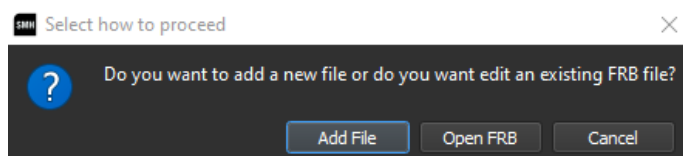
There is a simple way to program Option Bytes.

1. Select: Manually compose a new FRB file.

 Device Wizard



2. Select: Add File



3. On Fill Data / Variable Data insert the following numbers.

Values flashed in this area will be defined through **DYNMEMSET** command. If no dynamic data covers this area, Filling value defined below will be flashed. Please check out 'Programmer's Manual' documentation for more information.

Memory Map:

Type	Memory	Start Address	End Address	Addressing
F	Program ROM	0x00000000	0x000003FF	16 Bit Word
O	Option ROM	0x00040000	0x0004000F	16 Bit Word
E	Emulated EEPROM	0x00400000	0x0040001F	16 Bit Word

Switch to Byte Memory Map

Start Address (hex) Converted 16 BIT WORD : 0x00040000

Size (hex) Converted 16 BIT WORD : 0x00040010

Value (hex)

Fits into device area: O - Option ROM
 Start Address: 0x00080000 - Target Start Address: 0x00040000 [16 BIT WORD]
 End Address: 0x0008001F - Target End Address: 0x0004000F [16 BIT WORD]

OK Cancel

4. Clicking Ok should take you on a screen like this.

Device Wizard

Advanced FRB Setup
Select a FRB file to use for device programming, or create a new one

Reset FRB Open FRB Add Edit Crop Remove Duplicate

Data Type	File Name	Type	Block Start Address	Block End Address	Block Encoding	Memory Area
FILL/VAR	Fill Value: 0x00	Block	0x00080000	0x0008001F	BYTE	[O] - Option ROM

5. Remember to add also binary of **Program ROM** or **Emulated EEPROM** in order to program the device properly. (Optional)

Device Wizard

Advanced FRB Setup
Select a FRB file to use for device programming, or create a new one

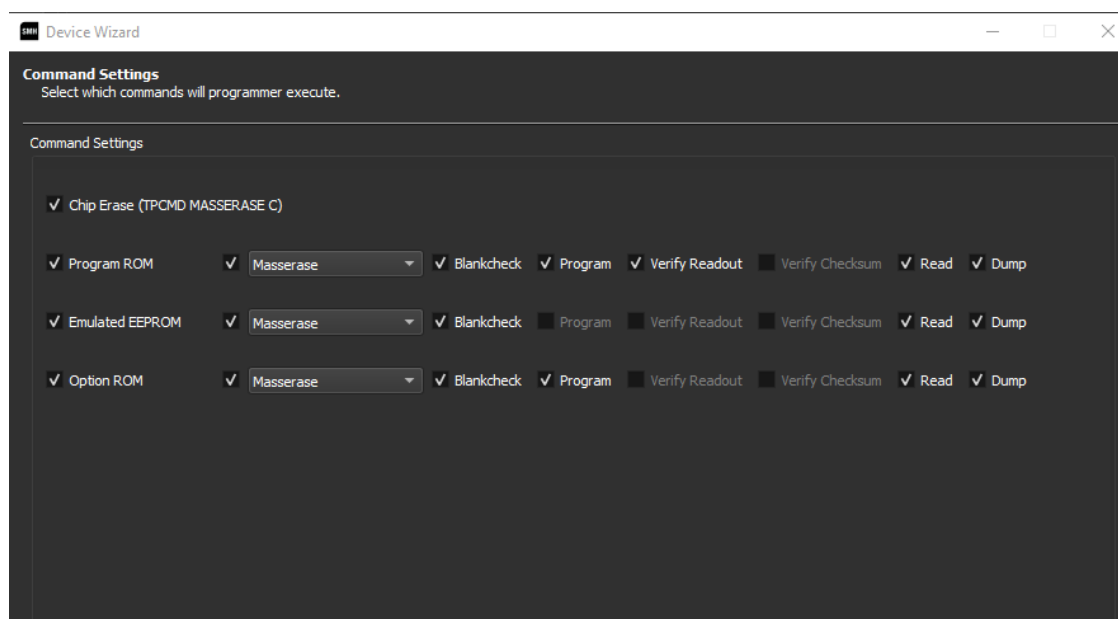
Reset FRB Open FRB Add Edit Crop Remove Duplicate

Data Type	File Name	Type	Block Start Address	Block End Address	Block Encoding	Memory Area
FILL/VAR	Fill Value: 0x00	Block	0x00080000	0x0008001F	BYTE	[O] - Option ROM
DATA	HT66F00x1.bin	Binary	0x00000000	0x000007FF	BYTE	[F] - Program ROM

6. Now you will be able to select Program on Option ROM. Select only operation that you need.

Note: since Option Bytes are calculated in runtime by algorithm there is no need to compare their values with FRB file data, so Verify is unnecessary, Read Option ROM instead.

That's it!



5. Security Management

HT66F00x1 flash technology

Use this command to **lock** the device.

✓ TPCMD LOCK

```
#TPCMD LOCK
```

Remember that the only way to **unlock** the device is to perform a Masserase C command.
Please note that this command will erase all data on the device, so take care to place the program command below in order to insert the data back into the device.