

DediProg[®] EM100 SPI Flash Emulator Presentation



May 2010

EM100 features

- ***Emulate all the market Serial Flash***
- ***1.8-3.6V, up to 75MHz, 512Kb to 128Mb***
- ***The Highest Code download performances: less than 4 seconds for code update whatever the density selected***
- ***Display and Edit memory content evolution***
- ***Debug functions: SPI Trace, SPI Trigger on SPI events, SPI Hyper Terminal, Memory status...***
- ***No need to remove the Serial Flash soldered on board***

“Simply the best solution to develop code on SPI Flash memories”

Serial Flash Emulation



DediProg

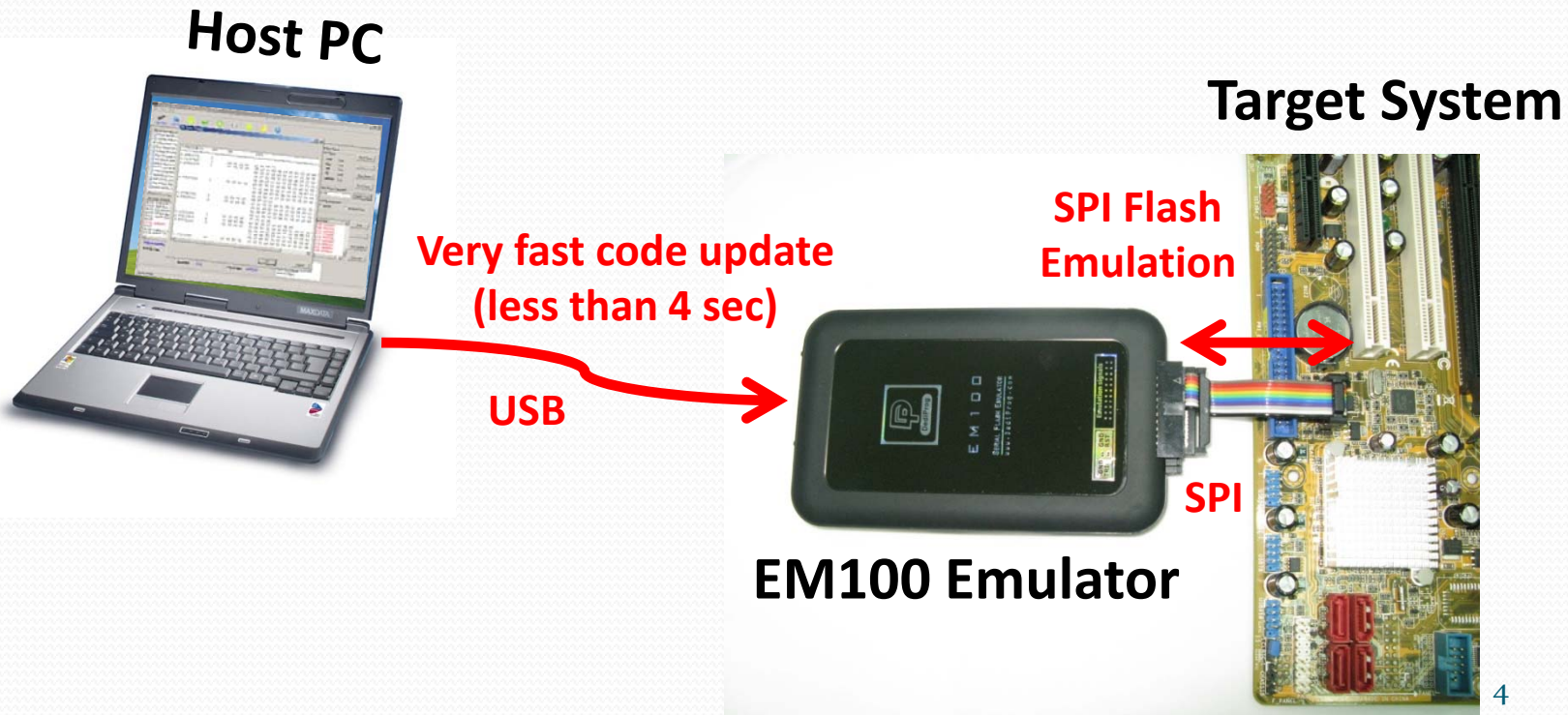
EM100 benefits

Accelerates embedded software development

EM100 provides high speed links from host to target via USB2 interface allowing very fast code download. EM100 completely eliminates the delays in Erasing and programming memories or loading code to target memory.

“minutes become seconds”

“Bios engineers save between up to 1 hour every days”



Save time when updating code

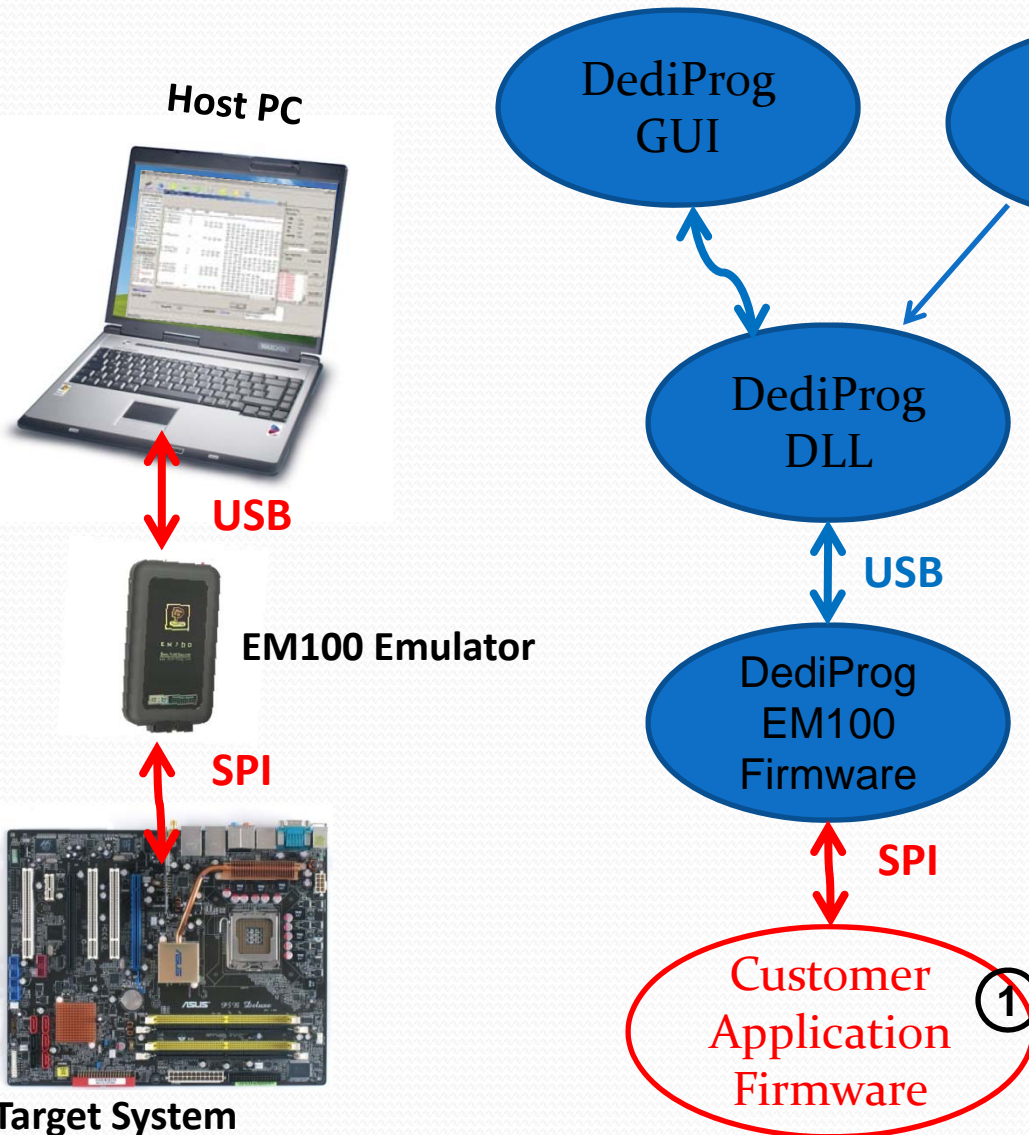
The EM100 SPI Flash Emulator can help engineers saving time When multiple SPI Flash update are required for development. The table below highlight the advantages of using the EM100 rather Than a Real SPI Flash even if updated with the high performance SF100 programmer.

Densities	Erase + Program + Verify	
	Real SPI Flash*	EM100
SPI Flash 512Kb	>3 seconds	< 3 seconds
SPI Flash 1Mb	> 4 seconds	< 3 seconds
SPI Flash 2Mb	>6 seconds	< 3 seconds
SPI Flash 4Mb	>12 seconds	< 3 seconds
SPI Flash 8Mb	>18 seconds	< 4 seconds
SPI Flash 16Mb	>25 seconds	< 4 seconds
SPI Flash 32Mb	>40 seconds	< 4 seconds
SPI Flash 64Mb	>90 seconds	< 4 seconds
SPI Flash 128Mb	>190 seconds	< 4 seconds

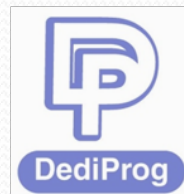
** Updating time of real SPI Flash with the high performance SF100.*

Two user interface: GUI and CML

- A user friendly Graphic Interface (easy to use)
- A Convenient Command Line Interface (controlled by user or software)



① Application Firmware can boot from the SPI Flash emulator. By adding a macro, the application firmware can also send debug messages to the HOST PC. Sample code can be asked to DediProg.



Serial flash Emulation

1: Select a Serial Flash memory to be emulated

4: Run Emulation

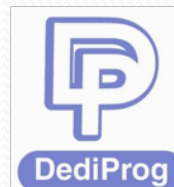
6: Edit memory content

7: All operations in one click

The screenshot shows the DediProg EM100 serial flash emulator 3.1.01 interface. The top toolbar contains icons for CHIP SELECT, OPEN FILE, DOWNLOAD, VERIFY, RUN, STOP, EDIT, BATCH, and CONFIGURE. The main window is divided into several sections: an EM100 Operation Log on the left, SPI Bus Status on the right, and a SPI Hyper Terminal at the bottom. The bottom section contains four panels: Hold Pin Setting (Default Low), Memory Info (Type: M25P128, Manufact.: Numonyx, Size(KB): 16384), File Info (Name: 8R.bin, Size: 0x100000 Bytes, Checksum: 00006968), and Batch Config Setting (Stop Emulation, Reload file, Download to EM100, Verify from EM100, Start Emulation). A status bar at the bottom left shows 'Device Ready'.

Annotations on the screenshot:

- 1: Select a Serial Flash memory to be emulated (points to CHIP SELECT)
- 2: Load the file (points to OPEN FILE)
- 3: Download the code in the EM100 in few seconds (points to DOWNLOAD)
- 4: Run Emulation (points to RUN)
- 5: Stop Emulation (points to STOP)
- 6: Edit memory content (points to EDIT)
- 7: All operations in one click (points to BATCH)



Debugging Features



DediProg

Application debugging

SPI Flash Pins status

SPI Bus trace

Last SPI Command issued

Application debug messages through our SPI Hyper Terminal

File View Help

CHIP SELECT OPEN FILE DOWNLOAD VERIFY RUN STOP EDIT BATCH CONFIGURE

EM100 Operation Log

- Downloading, please wait ...
- Download Complete
- EM100 Hold-Pin Setting: Default Low
- Checking Authentication....
- Authentication Pass
- Start EM100. EM100 is in Emulation mode. Ready to boot your system now.
- Stop EM100. EM100 in STOP mode now.
- EM100 Hold-Pin Setting: Floating by default
- EM100 Hold-Pin Setting: Default Low
- Checking Authentication....
- Authentication Pass
- Start EM100. EM100 is in Emulation mode. Ready to boot your system now.
- Stop EM100. EM100 in STOP mode now.
- EM100 Hold-Pin Setting: Floating by default

SPI Bus Status

Pin Status

CS#	Low
CLK	Low
SO	Low
SI	Low
HOLD#	Low

Last Issued Command

0x05

System Boot From:

EM100 On Board Chip

Start Trace

Stop Trace

Clear Buffer

Save Trace

Display Trace

SPI Hyper Terminal

Check root Start

Stop

Clear Buffer

Save Log

Hold Pin Setting	Memory Info	File Info	Batch Config Setting
Default Low	Type: M25P128 Manufact.: Numonyx Size(KB): 16384	Name: 8R.bin Size: 0x100000 Bytes Checksum: 00006968	Stop Emulation Reload file Download to EM100 Verify from EM100 Start Emulation

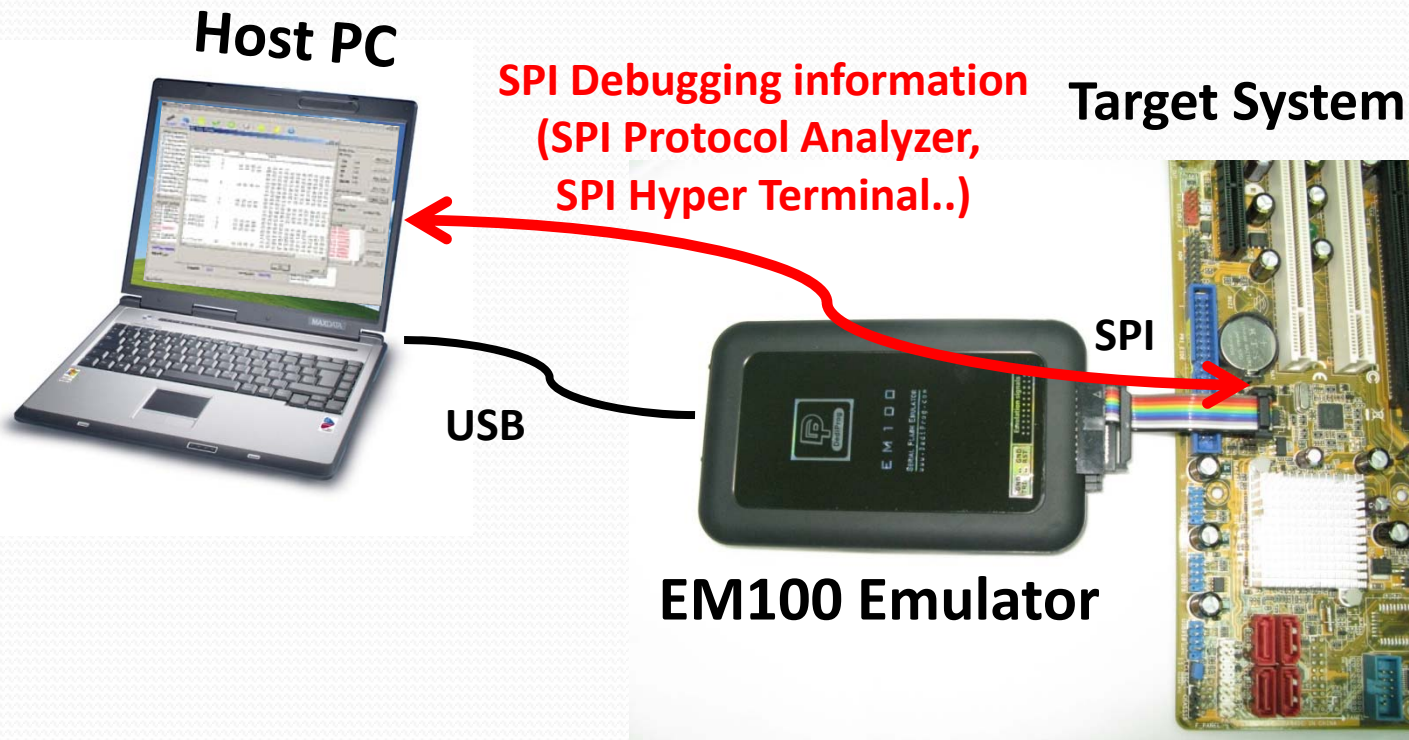
Device Ready

EM100 debugging benefits

Powerful software debugging features

EM100 improves debugging performance:

- Engineer can test his system on any market Serial Flash just in one click
- Engineer can analyze all the SPI bus communication and filter it
- Engineer can customize his own debugging method and messages by sending debug messages to the Host PC through the SPI bus.



SPI Trace

EM100 provides the “SPI Trace” function which can record the SPI communication on the Bus. Engineers will be able to monitor what happens on the application SPI bus and debug at a very low level. The SPI history can be displayed and saved in a file. More features will be added to decode the SPI bus command, research some specific command or filter the command recorded.

TIMESTAMP (ns)	CS#	CMD	DATA
5.99876773	1	03	ff ff ff ff
6.01087480	2	03	00 00 00 00
6.47831227	3	03	ff ff c0 e9 a4 fc 8d a4 24 00 00 00 00 8d d2 07 fc ff 10 00 8d a4 24 00 00 00 0f 09 e9 23 ff ff ff 00 00 20 00 6c 02 6d 02 6e 02 6f 02 fa b0 01 8e d8 be f0 ff 80 3c ea 75 05 ea 02 e6 80 66 2e 0f 01 16 a8 ff 0f 0f 09 e9 23 ff 02 50 02 58 02 59 02 68 02 69 6c 02 6d 02 6e 02 6f 02 fa b0 01 8e d8 be f0 ff 80 3c ea 75 05 ea 2 e6 80 66 2e 0f 01 16 a8 ff 0f 0f 09 e9 23 47 00 60 ff 4f 00 60 ff ff ff ff 8b ff 8e d8 be f0 ff 80 3c ea 75 05 ea 02 e6 80 66 2e 0f 01 16 a8 ff 0f 0f 09 e9 23 22 c0 f0 b8 08 00 8e d8 8e c5 8e c5
6.47864163	4	03	ff ff 00
6.47867722	5	03	ff ff f0
6.47868136	6	03	ff ff 00
6.47871522	7	03	ff ff a8
6.47871934	8	03	ff ff a8
6.47872530	9	03	ff ff 00
6.47875767	10	03	ff ff f0

Instruction Timing information According to the SPI Trace start

Instruction counter

SPI command operating code (Read, Fast read..)

SPI Address

SPI Data



SPI Hyper Terminal

For more information on the SPI Hyper Terminal, please refer to our SPI Hyper Terminal Specification



SPI Hyper Terminal benefits

Debug easily your boot process

The EM100 offers the possibility to display information coming from the application firmware through the SPI bus available.

Benefits:

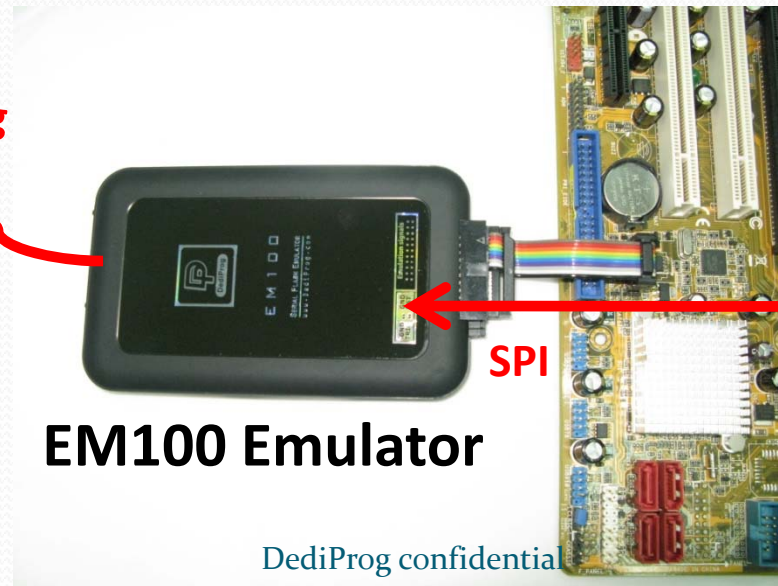
- *Application do not need to add an extra bus for debug purpose only (like JTAG..).*
- *Engineers do not need to wait the availability of the debug bus (like USB...) to start debugging. The SPI bus is available at the very beginning of the boot.*
- *Each engineers can customize the debug messages according to their needs*

Host PC



USB

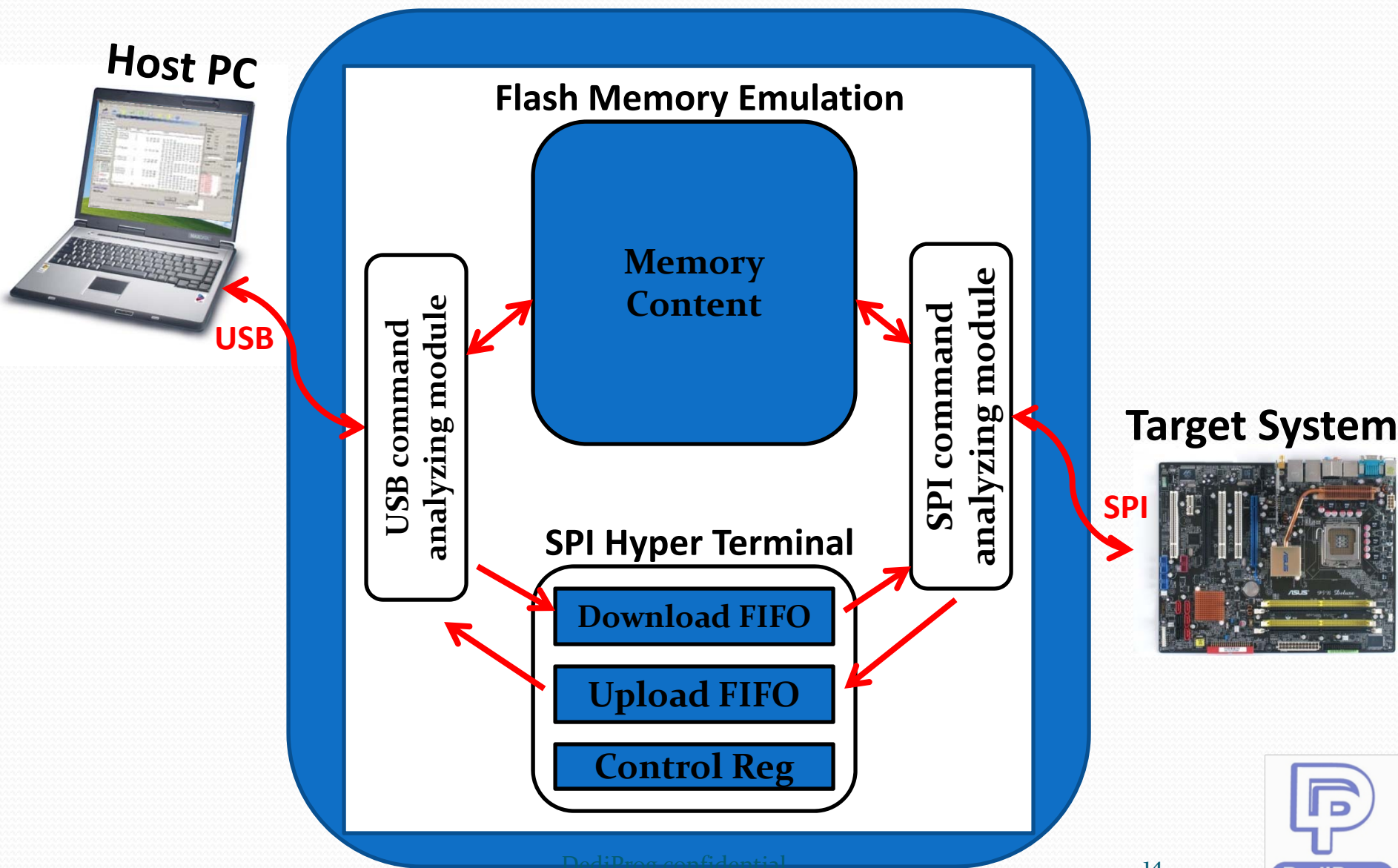
Target System



Application firmware sends information on the process on going through the SPI bus (checkpoints, ASCII messages..)

SPI Hyper Terminal architecture

EM100 Emulator



SPI Hyper Terminal resources

FIFO resources:

dFIFO	64Bytes	Used to transfer information from Host PC to application (EM100 Pro)
uFIFO	512 Bytes	Used to transfer information from application to Host PC

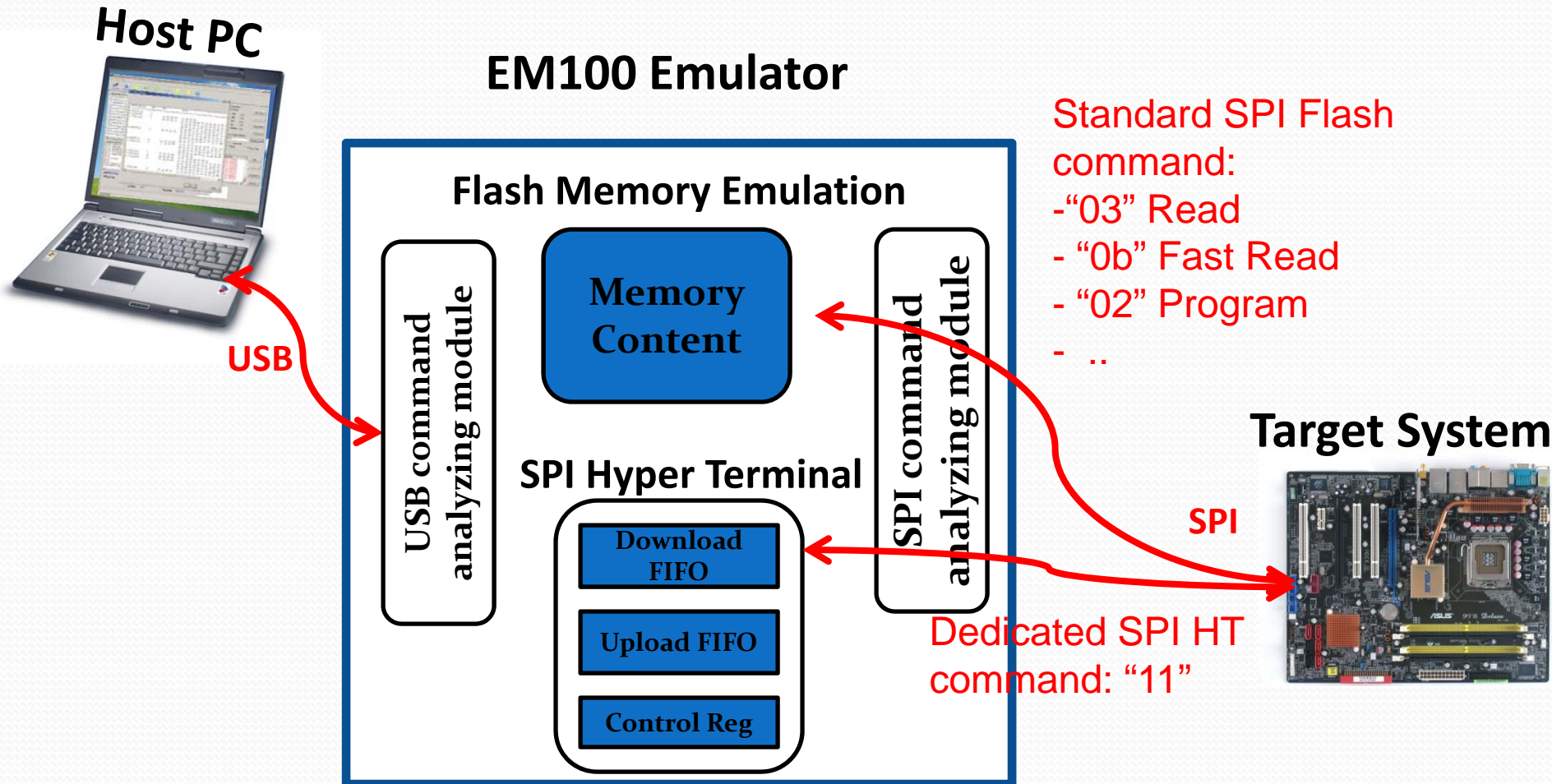
Registers resources:

Register 0	1 Byte	R/W	FIFO Overflow, pause/start emulation, FIFO valid data..
Register 1	1 Byte	R	Length of Valid data in the dFIFO
Register 2	1 Byte	R	Length of Valid data in the uFIFO
Register 3	1 Byte	R	EM100 identification on the SPI bus to differentiate from real SPI Flash
Register 4	1 Byte	R/W	uFIFO data format to indicate to the software how to read the data
Register 5	4 Bytes	R/W	EM100 timer to time stamp the debug message

Application firmware Communication with SPI Hyper Terminal



Dedicated SPI command 1/2



Dedicated SPI command 2/2

Instruction	Byte1	Byte 2	Byte 3	Byte 4	Byte 5	Byte n
Write Reg	11h	Don't care	AXh *	Reg value	None	None
Read Reg	11h	Don't care	BXh *	Don't care	Reg value	0
Write uFIFO	11h	Don't care	C0h	uFIFO Data	uFIFO Data	uFIFO Data..
Read dFIFO	11h	Don't care	D0h	Don't care	dFIFO Data	dFIFO Data..

* X defined the register address

Example: Write 10h to register 0

	Byte 1	Byte 2	Byte 3	Byte 4	Byte5	Byte n
Application MOSI	11h	Don't care	A0h	10h	Don't care	Don't care
EM100 MISO *	HZ	HZ	HZ	HZ	HZ	HZ
SPI Flash MISO **	HZ	HZ	HZ	HZ	HZ	HZ

Example: Read register 1 content

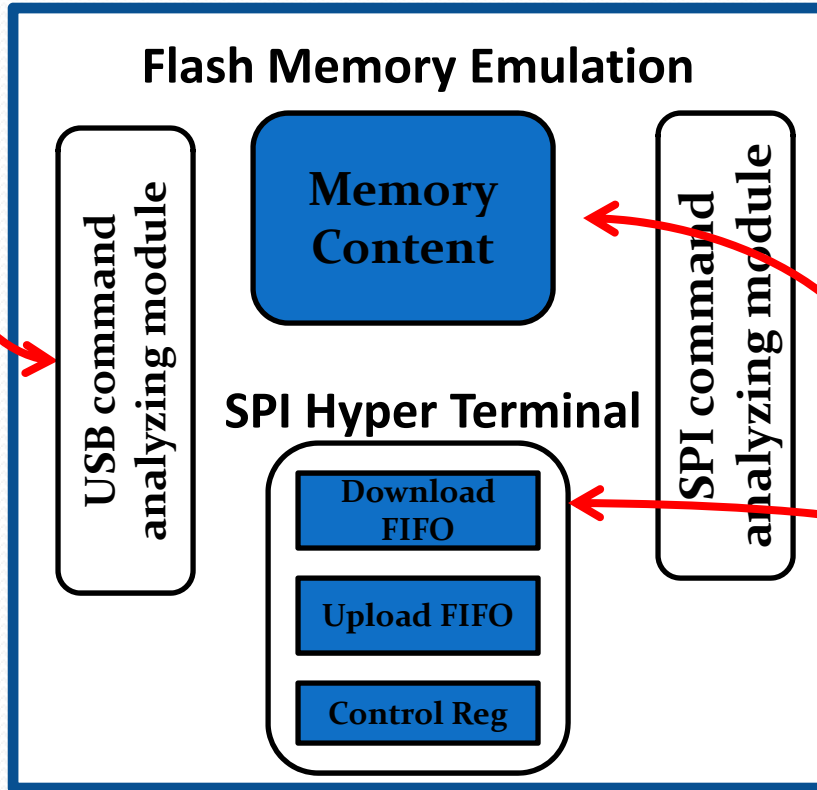
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 5
Application MOSI	11h	Don't care	B1h	Don't care	Don't care	HZ
EM100 MISO *	HZ	HZ	HZ	HZ	Data	HZ
SPI Flash MISO **	HZ	HZ	HZ	HZ	HZ	HZ

Standard Read command 1/2

Host PC



EM100 Emulator



Standard SPI Flash command:

- "03" Read
- "0b" Fast Read
- "02" Program
- ..

Target System



Standard SPI Flash read command with sequence of specific address:

- 03 XX AA A0 ..
- 03 XX 55 50 ..
- 03 XX AA A0 ..
- ➔ SPI HT

Dedicated SPI command 2/2

Enter the SPI Hyper Terminal mode with successive Read sequences

	Opcode	Address			
	Byte1	Byte 2	Byte 3	Byte 4	Byte n
Read sequence 1	03h	Don't care	AAh	A0h	Don't care
Read sequence 2	03h	Don't care	55h	50h	Don't care
Read sequence 3	03h	Don't care	AAh	A0h	Don't care
Read sequence 4	03h	SPI Hyper terminal command			
Read sequence n	03h	SPI Hyper terminal command			
End sequence	03h	Don't care	E0h	0	Don't care

SPI Hyper Terminal commands

Read Sequence	Instruction	Byte1	Byte 2	Byte 3	Byte 4	Byte 5
4+n	Write Reg	03h	Don't care	AX	Reg value	None
4+n	Write uFIFO	03h	Don't care	C0	uFIFO Data	None
End	Quit	03h	Don't care	E0	0	None

SPI Hyper Terminal Data format

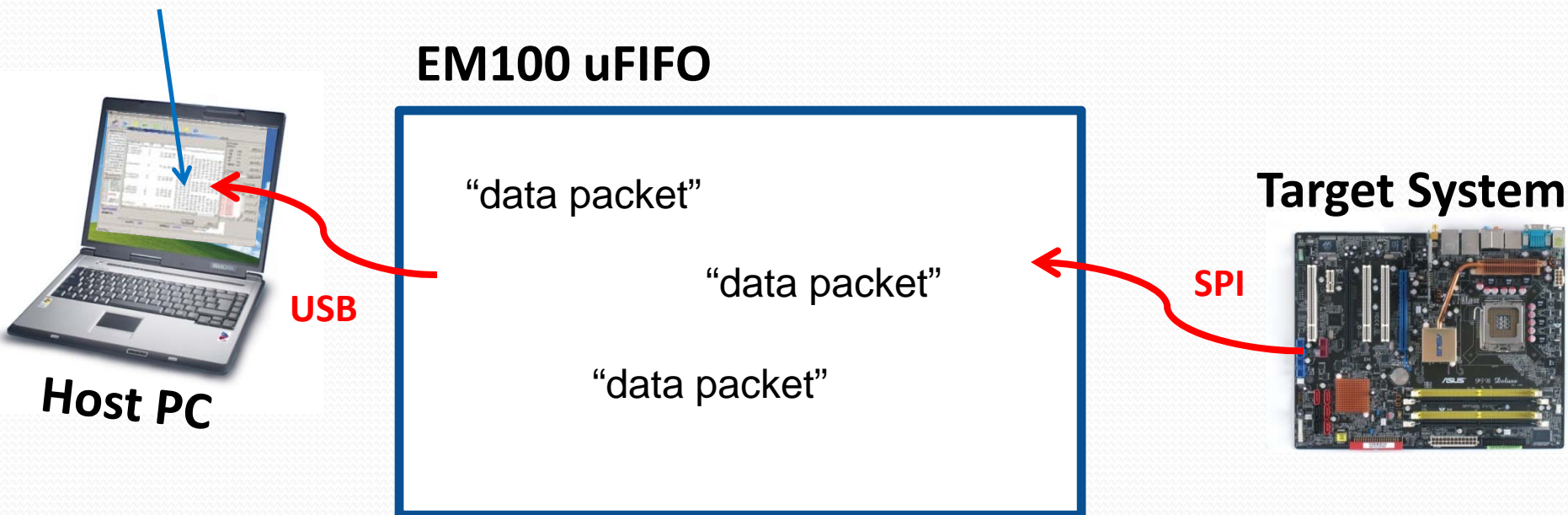


DediProg

SPI HT data format

A data format must be defined between the application firmware and the Host PC software to understand the meaning of the data transfer through the EM100 FIFO. DediProg has defined a standard format of our own GUI display as described in the next slides.

Debug messages
are displayed according
To the data format defined



Mixed Data Format

A data format must be defined between the application firmware and the Host PC software to understand the meaning of the data transfer through the EM100 FIFO.

Packet Format:

	Packet Header			Data
	Byte0-3	Byte4	Byte5	Byte6 to Byte263
Meaning	Packet Start Identification DWORD	Data Type in the packet	Data length	Data bytes from application firmware
Content	0x40, 0x44, 0x36, 0x47	01h - 03h = CKP 04h = Hex 05h = ASCII 06h = Time Stamp 07h = Look up table	0 to 255	Data

Data type	meaning
01	1 Byte Checkpoint
02	2 Bytes Checkpoint
03	4 Bytes checkpoint
04	Hexadecimal data
05	ASCII data
06	Time stamp provided by EM100 to control the boot process in the time
07	Application sends index number (2 Bytes) to point a debug message in the Host PC look up table and display the given message. Extra ASCII can be also added.

Example: Checkpoints

The Application Firmware is sending 6 Checkpoints to Host PC:

Data Format:

- Start identification: 40 44 36 47
- Data Type: 01 (Byte Checkpoint)
- Data length: 06 (6 checkpoints)
- Data: 01 02 03 04 05 06

SPI command:

- 11h XXh C0h 40h 44h 36h 47h 01 06 01 02 03 04 05 06

↑
SPI Hyper Terminal
Dedicated command

↑
Write to uFIFO

EM100 uFIFO

40h 44h 36h 47h 01 06 01 02 03 04 05 06



Host PC

Target System



SPI

Example: Hexadecimal values

The Application Firmware is sending 7 Hexadecimal values to Host PC:

Data Format:

- Start identification: **40 44 36 47**
- Data Type: **04 (Hexadecimal)**
- Data length: **07 (7 Hexadecimal)**
- Data: **01 AA 55 5F 6E 7B 8D**

SPI command:

- **11h XXh C0h 40h 44h 36h 47h 04 07 01 AA 55 5F 6E 7B 8D**

SPI Hyper Terminal
Dedicated command

Write to uFIFO

EM100 uFIFO

40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h
36h 47h 04 07 01 AA 55 5F 6E 7B 8D



Host PC

Target System



SPI

Example: ASCII messages

The Application Firmware is sending 8 ASCII characters to Host PC:

Data Format:

- Start identification: 40 44 36 47
- Data Type: 05 (ASCII)
- Data length: 08 (8 ASCII characters)
- Data: 50 4F 57 45 52 20 4F 4B

SPI command:

- 11h XXh C0h 40h 44h 36h 47h 05 08 50 4F 57 45 52 20 4F 4B

SPI Hyper Terminal
Dedicated command

Write to uFIFO

EM100 uFIFO

```
40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h  
36h 47h 04 07 01 AA 55 5F 6E 7B 8D 40h 44h 36h  
47h 05 08 50 4F 57 45 52 20 4F 4B
```



Host PC

Target System



SPI

Example: Time stamp 1/3

The Application Firmware is sending time stamp to Host PC in order to manage the boot timing:

Data Format:

- Start identification: 40 44 36 47
- Data Type: 06 (Time Stamp)
- Data length: 04 (4 Bytes)
- Data: B3 B2 B1 B0 (from EM100 Timer)

SPI command 1:

- 11h XXh C0h 40h 44h 36h 47h 05 08

↑
SPI Hyper Terminal
Dedicated command

↑
Write to uFIFO

EM100 uFIFO

```
40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h  
36h 47h 04 07 01 AA 55 5F 6E 7B 8D 40h 44h 36h  
47h 05 08 50 4F 57 45 52 20 4F 4B 40h 44h 36h  
47h 06 08 (time data are still missing)
```



Host PC

Target System



SPI

Example: Time stamp 2/3

The Application Firmware copies the EM100 timer value to the EM100 uFIFO to fill the missing time Stamp data:

Data Format:

- Start identification: 40 44 36 47
- Data Type: 06 (Time Stamp)
- Data length: 04 (4 Bytes)
- Data: B3 B2 B1 B0 (from EM100 Timer)

SPI command 2:

- 11h XXh B5h XXh B3h B2h B1h B0h

SPI Hyper Terminal
Dedicated command

Read EM100
Register 5

Timer value is output on the SPI
bus but also automatically copied
to the uFIFO

EM100 Timer

B3 B2 B1 B0

EM100 uFIFO

40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h
36h 47h 04 07 01 AA 55 5F 6E 7B 8D 40h 44h 36h
47h 05 08 50 4F 57 45 52 20 4F 4B 40h 44h 36h
47h 06 04 B3 B2 B1 B0



Host PC

Target System

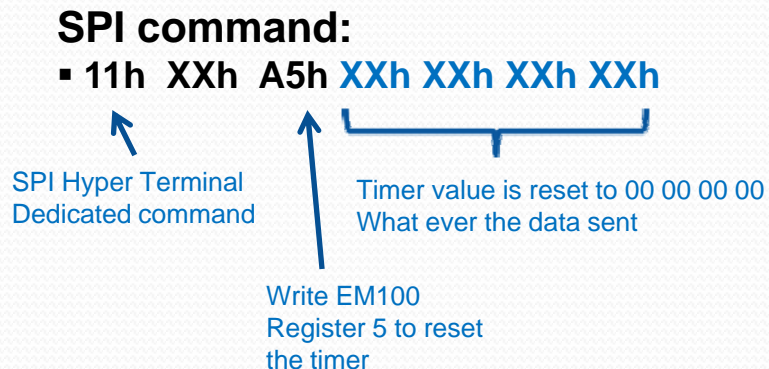


SPI

Example: Time stamp 3/3

The Application Firmware can have previously reset the EM100 timer to set the time reference (could be at the beginning of the boot).

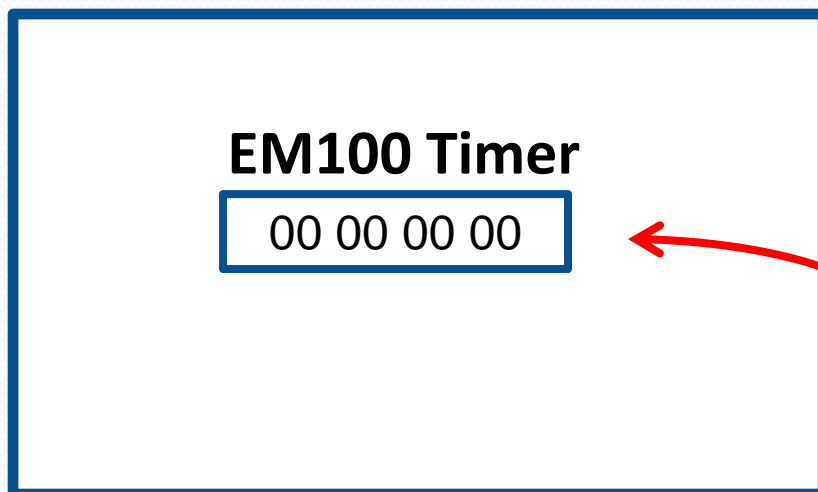
The EM100 timer is a 4 Bytes hexadecimal timer which is incremented every 10ns.
The DediProg software will display the timing In seconds.



EM100



Host PC



Target System



SPI

Example: Look up table 1/2

The application firmware will send a simple indexing parameter of two hexadecimal bytes to the EM100 uFIFO which will be used by the Host PC software to display its associated debugging messages located in the look-up table previously loaded by user in the host PC. The rest of the data will be in ASCII format to add information (like variable value, etc...) after the message. This data format will reduce the number of information to be sent by the application firmware to the Host PC through the SPI bus as they are predefined.

Look up table previously loaded in the software:

[DediProg]

0000=Value of the data register1:

0001=Value of the data register2:

0002=Value of the data register3:

....

Select the message
According to the index

②

EM100

①

Send Index
and ASCII to
EM100

Target System

SPI

Index
(2 Bytes)

③

Display the message
Selected in the look up
table

④

Display the additional ASCII
Characters to customize the
Messages (parameter value..)



Host PC



Example: Look up table 2/2

The Application Firmware is sending the look up table index and some additional ASCII characters:

Data Format:

- Start identification: 40 44 36 47
- Data Type: 07 (Look up table)
- Data length: 04 (2 bytes for Index + 2 ASCII)
- Data: 2 Bytes Index + 2 ASCII characters



EM100 uFIFO

```
40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h
36h 47h 04 07 01 AA 55 5F 6E 7B 8D 40h 44h 36h
47h 05 08 50 4F 57 45 52 20 4F 4B 40h 44h 36h
47h 06 04 B3 B2 B1 B0 40h 44h 36h 47h 07 04 00
01 31 32
```



Host PC

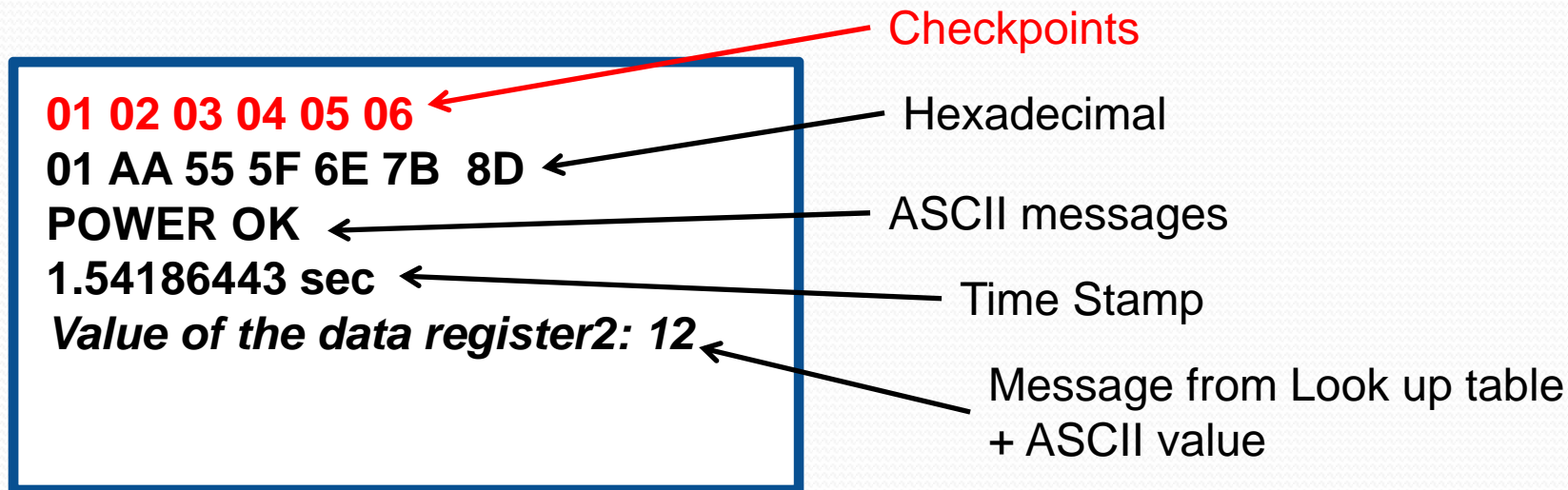
Target System



SPI

Example: Software display

The DediProg software on the Host PC will then display each debug messages according to their format:



EM100 uFIFO

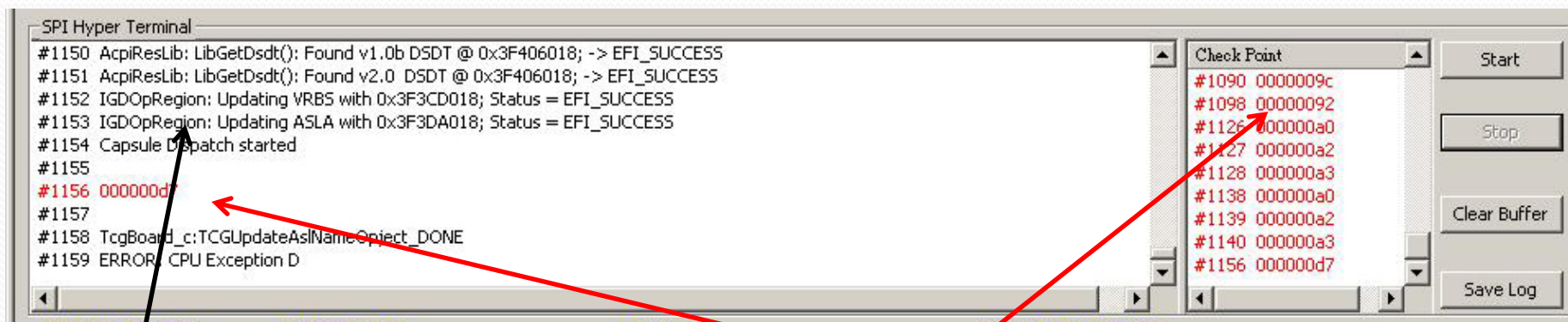
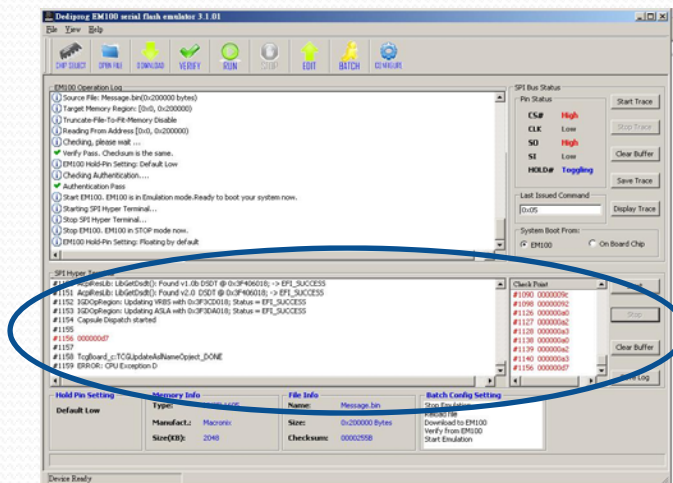
```
40h 44h 36h 47h 01 06 01 02 03 04 05 06 40h 44h  
36h 47h 04 07 01 AA 55 5F 6E 7B 8D 40h 44h 36h  
47h 05 08 50 4F 57 45 52 20 4F 4B 40h 44h 36h  
47h 06 04 B3 B2 B1 B0 40h 44h 36h 47h 07 04 00  
01 31 32
```

Target System



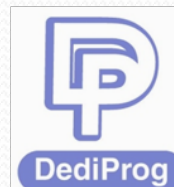
Host PC

Display: ASCII messages + checkpoints

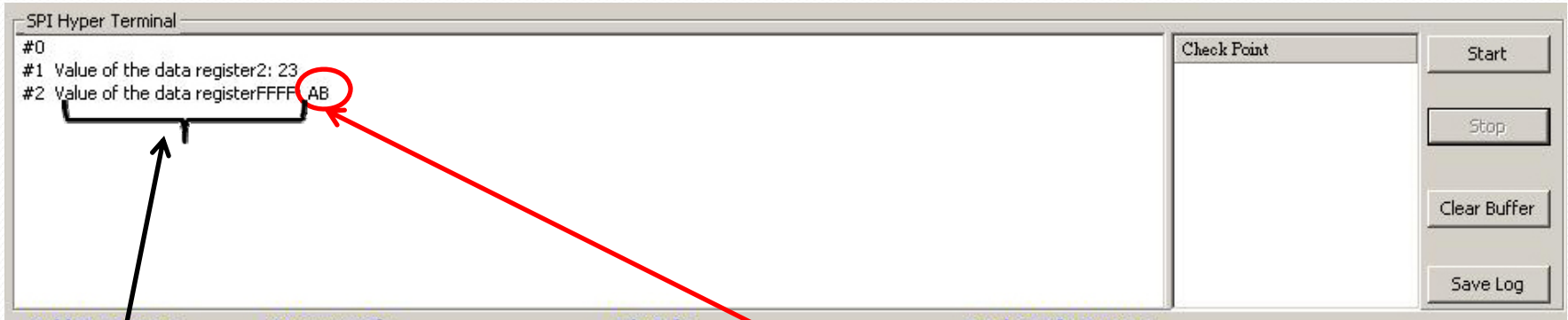
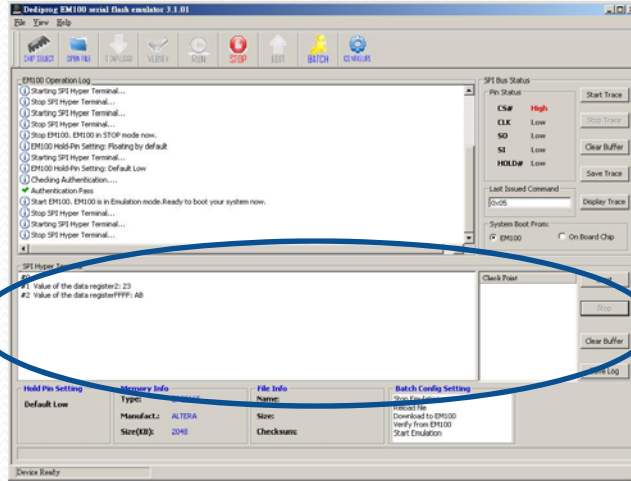


ASCII
messages

Checkpoints



Display: Look up Table messages + ASCII

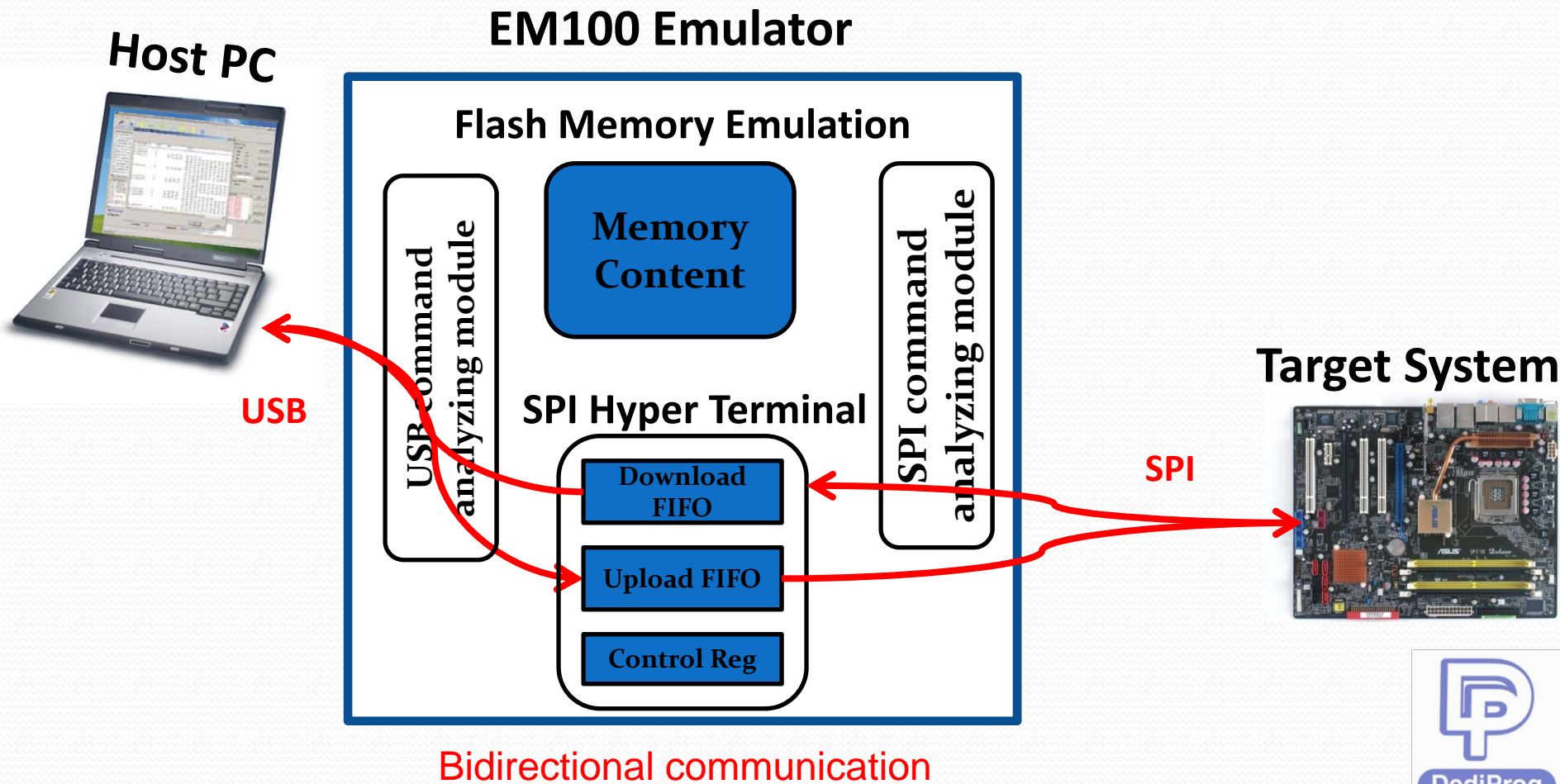


Look up table messages
Selected with the 2 bytes
Index in the uFIFO

ASCII to customized the look up table message
Example: register value, variable value..

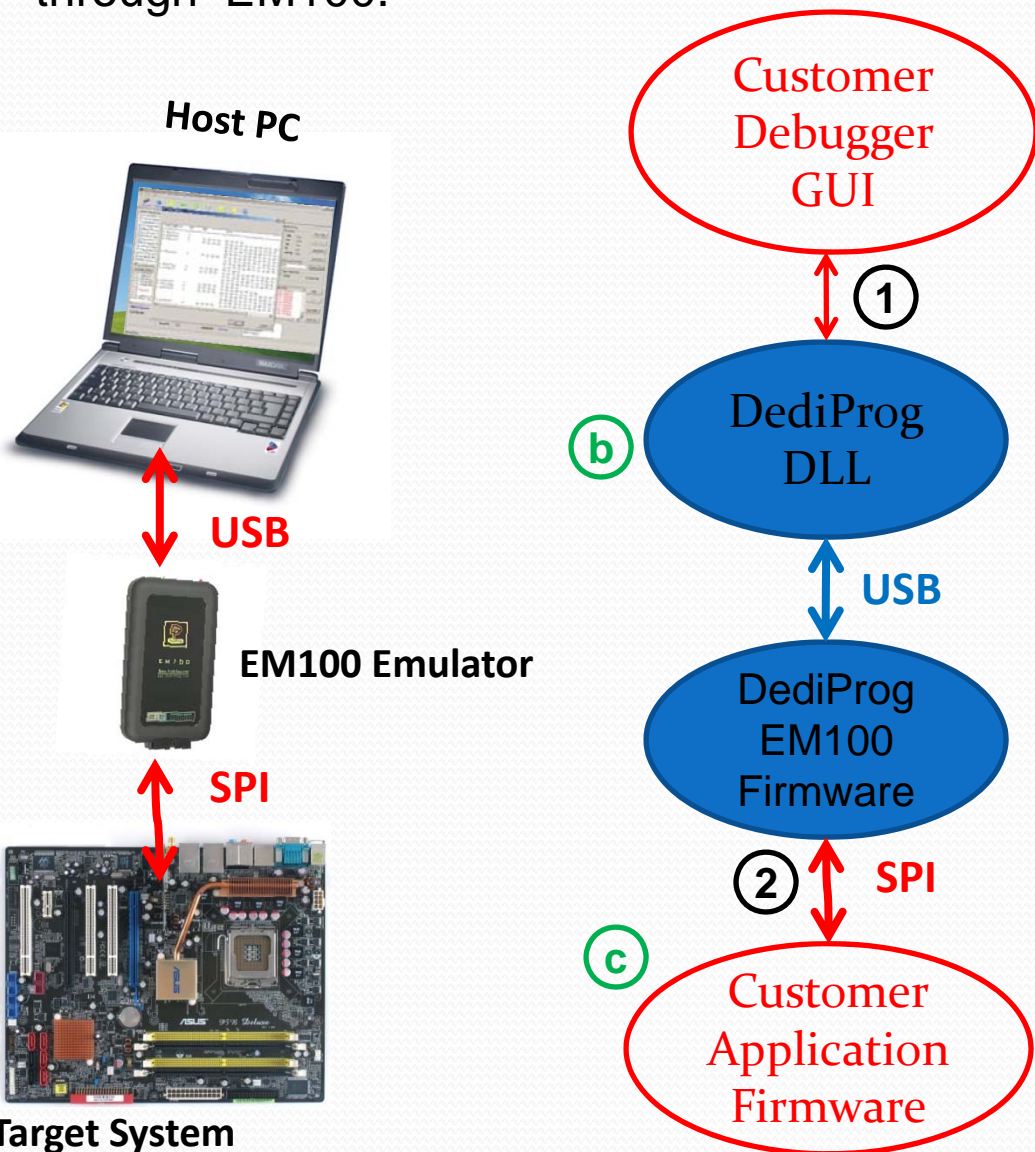
EM100 for Bidirectional debugger

Use EM100 and DediProg API to develop your own application / chip debugger.
PC Host software can control the application through USB and SPI bus !



Develop your own debugger

DediProg provides DLL for customers planning to develop their own debugger through EM100.



① Customer Debugger GUI can use DediProg DLL to configure and start the memory emulation and control the EM100 SPI Hyper Terminal bidirectional communication

② Application firmware will communicate with the GUI through SPI and EM100 FIFO.

① a DediProg can provide a sample code for the Customized GUI under request.

① b DediProg provides EM100 DLL under request.

① c DediProg can provide the application firmware sample code under request.

EM100 Pin out assignment

Output signals

3	1
GND	GND
Trig	Reset
4	2

Memory signals

19	17	15	13	11	9	7	5	3	1
GND	CTRL	CTRL	CTRL	3.3V	GND	WP1	MISO	CS1	CTRL
CTRL	CTRL	3.3V	NC	CTRL	MOSI	CLK	Hold 1	Vcc	CTRL
20	18	16	14	12	10	8	6	4	2

Reset output to reset the application and synchronize it with the EM100.

SPI Bus control

Hold pin can drive the on board Serial flash
To disable it (no need to remove it)

For more information on how to connect EM100 to your target board please refer to our “EM100 connection” presentation.