HE910/UE910/UL865 Families Ports Arrangements User Guide

1vv0300971 Rev.9 – 2015-02-16

Making machines talk.

APPLICABILITY TABLE

	SW Versions
HE910 Family	
HE910 [1]	**12.00.xx6**
HE910-D	**12.00.xx6**
HE910-EUR / HE910-EUD	**12.00.xx6**
HE910-EUG / HE910-NAG	**12.00.xx6**
HE910-NAR / HE910-NAD	**12.00.xx6**
UE/UL Family (Embedded)	
UE910-EUR / UE910-EUD	**12.00.xx6**
UE910-NAR / UE910-NAD	**12.00.xx6**
UL865-EUR / UL865-EUD	**12.00.xx6**
UL865-NAR / UL865-NAD	**12.00.xx6**
UL865-N3G	**12.00.xx6**

Note: the products equipped with the software versions equal or higher than the versions shown in the table support the features described in the present document. See also the "Document History" chapter.

SERVICES COEXISTENCE TABLE

	Services			
	Embedded GPS	External GPS	Python	AppZone
HE910 Family			**Python and AppZone are mutually exclusive**	
HE910	✓		✓	✓*
HE910-D		✓	✓	
HE910-EUR / HE910-EUD		✓	✓	
HE910-EUG / HE910-NAG	✓		✓	
HE910-NAR / HE910-NAD		✓	✓	
UE/UL Family (Embedded)				
UE910-EUR / UE910-EUD		✓	✓	✓*
UE910-NAR / UE910-NAD		✓	✓	✓*
UL865-EUR / UL865-EUD		✓	✓	
UL865-NAR / UL865-NAD		✓	✓	
UL865-N3G		✓	✓	

Note: the table summarizes the Services provided by the modules when they are equipped with the suitable software version, and shows the Services coexistence.

**(***): AppZone available on demand on specific part numbers.

[1] HE910 is the "type name" of the products marketed as HE910-G & HE910-DG

SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

LEGAL NOTICE

These Specifications are general guidelines pertaining to product selection and application and may not be appropriate for your particular project. Telit (which hereinafter shall include, its agents, licensors and affiliated companies) makes no representation as to the particular products identified in this document and makes no endorsement of any product. Telit disclaims any warranties, expressed or implied, relating to these specifications, including without limitation, warranties or merchantability, fitness for a particular purpose or satisfactory quality. Without limitation, Telit reserves the right to make changes to any products described herein and to remove any product, without notice.

It is possible that this document may contain references to, or information about Telit products, services and programs, that are not available in your region. Such references or information must not be construed to mean that Telit intends to make available such products, services and programs in your area.

USE AND INTELLECTUAL PROPERTY RIGHTS

These Specifications (and the products and services contained herein) are proprietary to Telit and its licensors and constitute the intellectual property of Telit (and its licensors). All title and intellectual property rights in and to the Specifications (and the products and services contained herein) is owned exclusively by Telit and its licensors. Other than as expressly set forth herein, no license or other rights in or to the Specifications and intellectual property rights related thereto are granted to you. Nothing in these Specifications shall, or shall be deemed to, convey license or any other right under Telit's patents, copyright, mask work or other intellectual property rights or the rights of others.

You may not, without the express written permission of Telit: (i) copy, reproduce, create derivative works of, reverse engineer, disassemble, decompile, distribute, merge or modify in any manner these Specifications or the products and components described herein; (ii) separate any component part of the products described herein, or separately use any component part thereof on any equipment, machinery, hardware or system; (iii) remove or destroy any proprietary marking or legends placed upon or contained within the products or their components or these Specifications; (iv) develop methods to enable unauthorized parties to use the products or their components; and (v) attempt to reconstruct or discover any source code, underlying ideas, algorithms, file formats or programming or interoperability interfaces of the products or their components by any means whatsoever. No part of these Specifications or any products or components described herein may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without the prior express written permission of Telit.

HIGH RISK MATERIALS

Components, units, or third-party products contained or used with the products described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (“High Risk Activities"). Telit, its licensors and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

TRADEMARKS

You may not and may not allow others to use Telit or its third party licensors’ trademarks. To the extent that any portion of the products, components and any accompanying documents contain proprietary and confidential notices or legends, you will not remove such notices or legends.

THIRD PARTY RIGHTS

The software may include Third Party Right software. In this case you agree to comply with all terms and conditions imposed on you in respect of such separate software. In addition to Third Party Terms, the disclaimer of warranty and limitation of liability provisions in this License shall apply to the Third Party Right software.

TELIT HEREBY DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED FROM ANY THIRD PARTIES REGARDING ANY SEPARATE FILES, ANY THIRD PARTY MATERIALS INCLUDED IN THE SOFTWARE, ANY THIRD PARTY MATERIALS FROM WHICH THE SOFTWARE IS DERIVED (COLLECTIVELY “OTHER CODE”), AND THE USE OF ANY OR ALL THE OTHER CODE IN CONNECTION WITH THE SOFTWARE, INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF SATISFACTORY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE.

NO THIRD PARTY LICENSORS OF OTHER CODE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND WHETHER MADE UNDER CONTRACT, TORT OR OTHER LEGAL THEORY, ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE OTHER CODE OR THE EXERCISE OF ANY RIGHTS GRANTED UNDER EITHER OR BOTH THIS LICENSE AND THE LEGAL TERMS APPLICABLE TO ANY SEPARATE FILES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Contents

Figures

Tables

1. Introduction

1.1. Scope

The present document provides a guideline to connect logically the physical interfaces (ports) of the module to the services supported by the module itself. It is up to the user to configure the module in suitable way to avoid hardware/software resources conflicts. The ports/services arrangement is any possible logical connection of a physical port to an available 'Access Point' (e.g. AT0, AT1, AT2, TT, PYSER, etc.) supported by the used module.

1.2. Audience

User Application designers may use this document to exploit at best the communication resources offered by the modules without run up against resources contentions among services.

1.3. Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at:

TS-EMEA@telit.com

TS-AMERICAS@telit.com

TS-APAC@telit.com

Alternatively, use:

http://www.telit.com/support

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.4. Related Documents

[1] Telit's CMUX Implementation User Guide, 1vv0300994
[2] HE910/UE910/UL865 AT Commands Reference Guide, 80378ST10091A
[3] Telit Easy Script Python, 80000ST10020a
[4] HE910 Hardware User Guide, 1vv0300925
[5] UE910 Hardware User Guide, 1vv0301012
[6] UL865 Hardware User Guide, 1vv0301050
[7] Telit's AppZone User Guide, 1vv0301082
[8] AppZone APIs User Guide, 1vv0301130

2. Ports Arrangements and Virtual Service Device

Before describing the several ports arrangements supported by the HE910/UE910/UL865 Families, refer to documents [4], [5] and [6], it is useful introduce the Virtual Serial Device.

Virtual Serial Device, hereafter called VSD, is a software layer designed to run on HE910/UE910/UL865 modules. It manages virtual connections between the physical serial ports, accessible to the user, and the services provided by the module. VSD supports several 'Access Points' used as anchorage points for the logical connections. Tab. 1 shows the physical and logical objects involved in the connections management: Physical Serial Ports, Software Access Points, AT Parsers and TT Utilities, Services, and Protocols.

Physical Serial Ports	Software Access Points	AT Parsers and TT Utilities	Services	Protocols
USIF0 USIF1 USB (USB0÷USB6)[2] SPI HSIC	AT0, AT1, AT2, AT3 TT DLink VHWDTE0 VHWDTE1 PYSER …… …… Python Debugging GPS	Instance #1 Instance #2 Instance #3 TTC, 3G	GPS Python AppZone	CMUX (VC1÷VC4)[3]

Tab. 1: Physical and Logical Objects Managed by VSD

NOTICE: in documents [4], [5], and [6] USIF0 and USIF1 are called respectively Modem Serial Port1 and Modem Serial Port 2

It is useful to remind the 'AT Command Parser Instances' and their relationships with the 'Access Points'. HE910/UE910/UL865 modules provide three 'AT Commands Parser Instances', which are logically independent and connected to three different 'Access Points'; each parser recognizes and executes the AT commands received on its 'Access Point'.

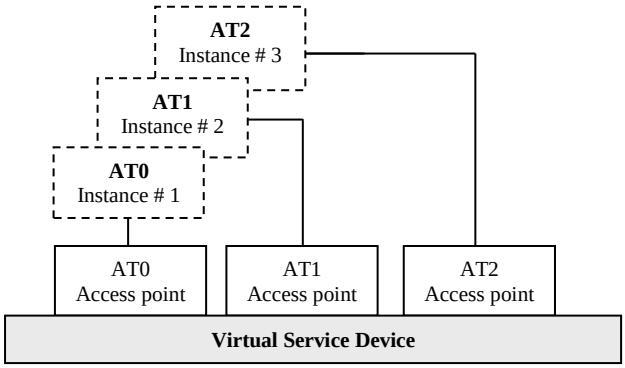

Fig. 1: AT Parser Instances

[2] Seven USB channels: USB0÷USB6.
[3] Four CMUX channels: VC1÷VC4.

3. Factory Ports Arrangements

NOTICE: as a rule, use the AT#PORTSCFG=? test command to have a short description of the supported ports arrangements in accordance with the software version installed on the module that you are using.

3.1. USB cable not plugged in

Let us assume that the module is using the factory setting[4] and USB cable is not plugged in. Now, power on the module. The Fig. 2 depicts the factory arrangement of the internal connections among physical ports and 'Access points'.

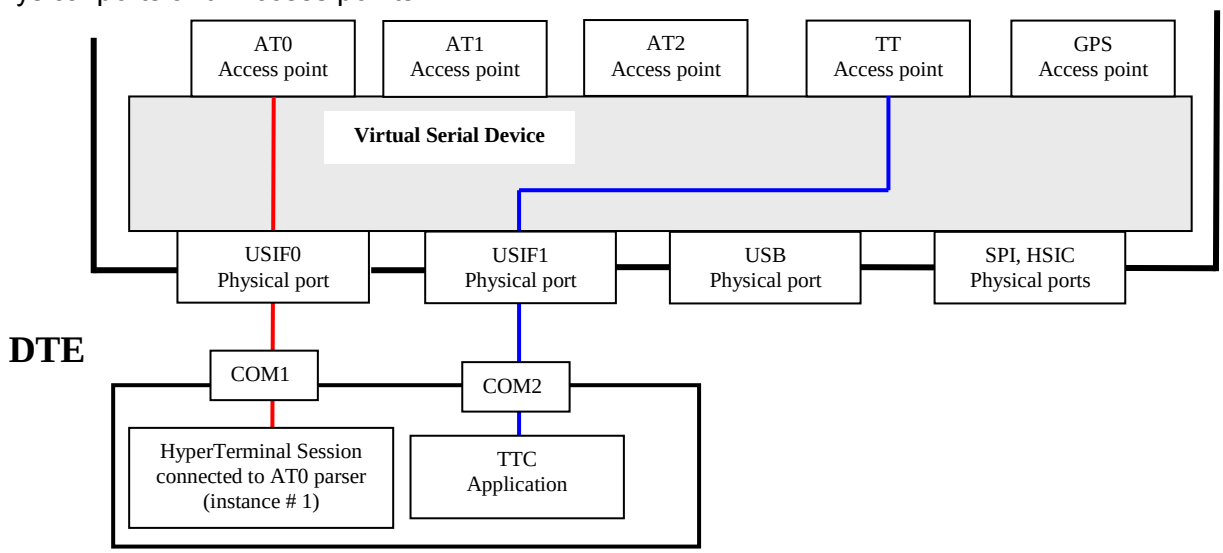

Fig. 2: Factory Ports Arrangement

Tab. 2 summarizes the factory ports arrangement.

AT#PORTCFG=1 (Factory setting)					
	AT0	**AT1**	**AT2**	**TT**	**GPS/NMEA**
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				
USIF1				TTC	
SPI					

[4] AT#PORTCFG=1, refer to Chapter 4 and document [2].

Tab. 2: Factory Ports Arrangement

NOTICE: the DTE used in the next examples is a Windows-PC and does not provide the SPI and HSIC interfaces. The following figures show SPI and HSIC interfaces in the same box when they are not involved in connections, but they are two distinct entities as shown in Fig. 2.

NOTICE: figures show the use of two types of trace applications in accordance with the selected configuration via AT#PORTCFG command, refer to chapter 4:

- TTC (Telit Trace Client tool);
- 3G tool (for internal use only).

3.2. USB cable plugged in

Assume that the module is powered on, and Fig. 2 shows its configuration[5]. Now, connect the USB cable to the module. The module recognizes the “plug in” event, and Fig. 5 shows its factory arrangement; Tab. 3 summarizes the new factory configuration. USB0÷USB6 are the seven channels of the USB port.

AT#PORTCFG=1 (Factory Setting)					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			
USB1					
USB2					
USB3			X		
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				
USIF1				TTC	
SPI					

Tab. 3: Factory Ports Arrangement with USB Cable

NOTICE: Telit provides the suitable USB drivers to install on Windows-PC; chapter 4 shows an example of mapping between USBx channels and virtual COMx ports.

[5] AT#PORTCFG=1, refer to Chapter 4.

4. AT#PORTCFG Command

AT#PORTCFG command manages several internal ports arrangements by means of its parameter value called 'Variant', refer to document [2]. The tables and figures reported on the next pages show the various ports configurations obtained changing the 'Variant' value of the command and/or connecting the USB cable to the module.

Here is the sequence to make active the entered AT#PORTCFG command:

- Assume to start from the configuration shown in Fig. 2, it is the factory setting: #PORTCFG is 1;
- Enter, for example, the AT#PORTCFG=0 command through USIF0 port, AT0 parser elaborates the just entered command, but no actions are taken;
- Power down the module;
- Power on the module. The AT#PORTCFG=0 command is executed and the ports arrangement of Tab. 5 is set.

The dialog box below shows an example of USBx channels ←→virtual COMx ports mapping on the DTE side; the mapping depends on the Windows-PC configuration.

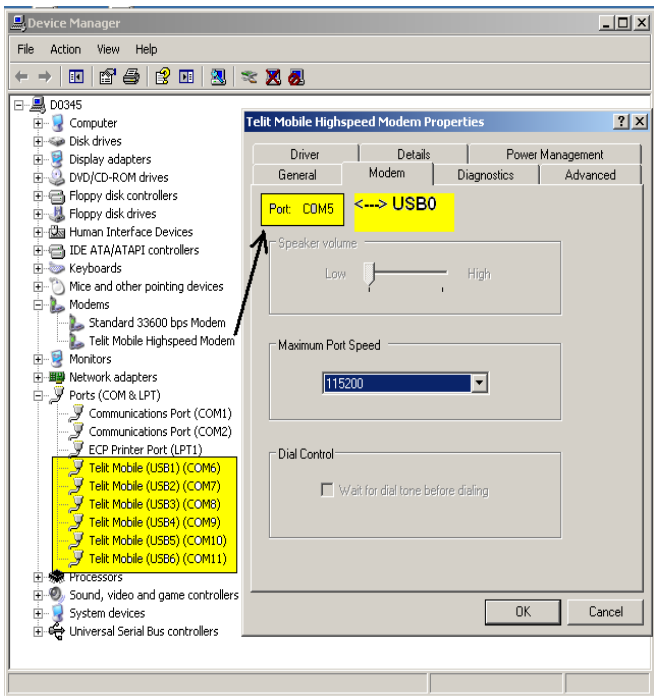

Fig. 3: USBx Channels Mapped into Virtual COMx Ports

From the dialog box shown on the lefts side is derived the next mapping table:

USB CHANNELS	VIRTUAL PORTS
USB0	VCOM5
USB1	VCOM6
USB2	VCOM7
USB3	VCOM8
USB4	VCOM9
USB5	VCOM10
USB6	VCOM11

Tab. 4: Mapping Table

Refer to chapter 5.2.

4.1. AT#PORTCFG=0

AT#PORTCFG=0					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1.1
USIF1					
SPI					

Tab. 5: #PORTCFG=0, no USB Cable

AT#PORTCFG=0					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1.2
USB1				TTC	
USB2					
USB3			X		Ref. chapter 6.1.1.2
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1.2
USIF1					
SPI					

Tab. 6: #PORTCFG=0, with USB Cable

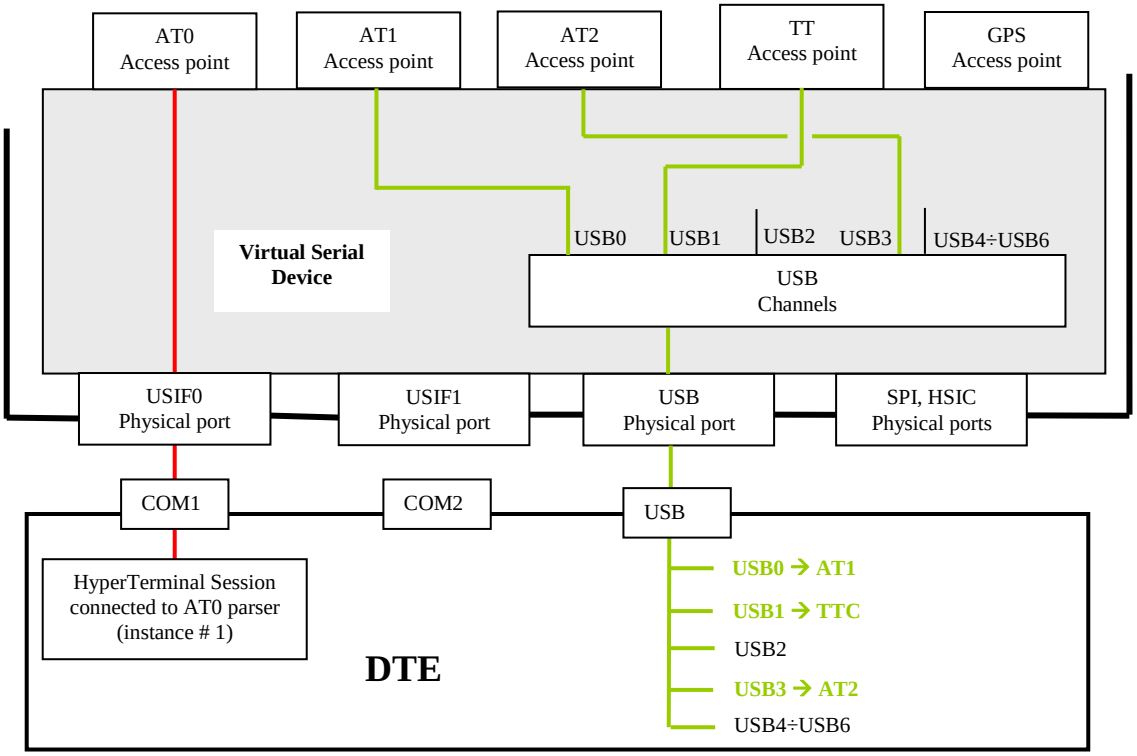

Fig. 4: #PORTCFG=0 + USB Cable

4.2. AT#PORTCFG=1

AT#PORTCFG=1 (Factory setting)					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1				TTC	
SPI					

Tab. 7: #PORTCFG=1, no USB Cable

AT#PORTCFG=1 (Factory setting)					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1					
USB2					
USB3			X		Ref. chapter 6.1.1
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1				TTC	
SPI					

Tab. 8: #PORTCFG=1, with USB Cable

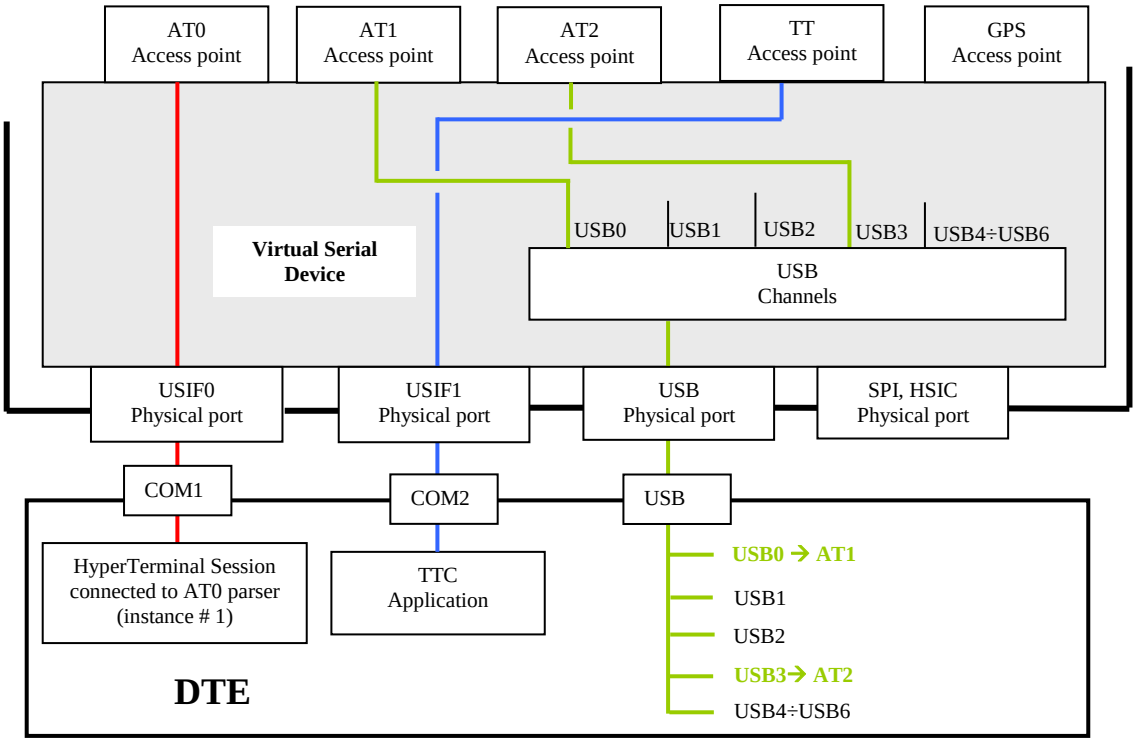

Fig. 5: #PORTCFG=1 + USB Cable

4.3. AT#PORTCFG=2

AT#PORTCFG=2					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI			X		Ref. chapter 6.1.1

Tab. 9: #PORTCFG=2, no USB Cable

AT#PORTCFG=2					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2					
USB3					
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI			X		Ref. chapter 6.1.1

Tab. 10: #PORTCFG=2, with USB Cable

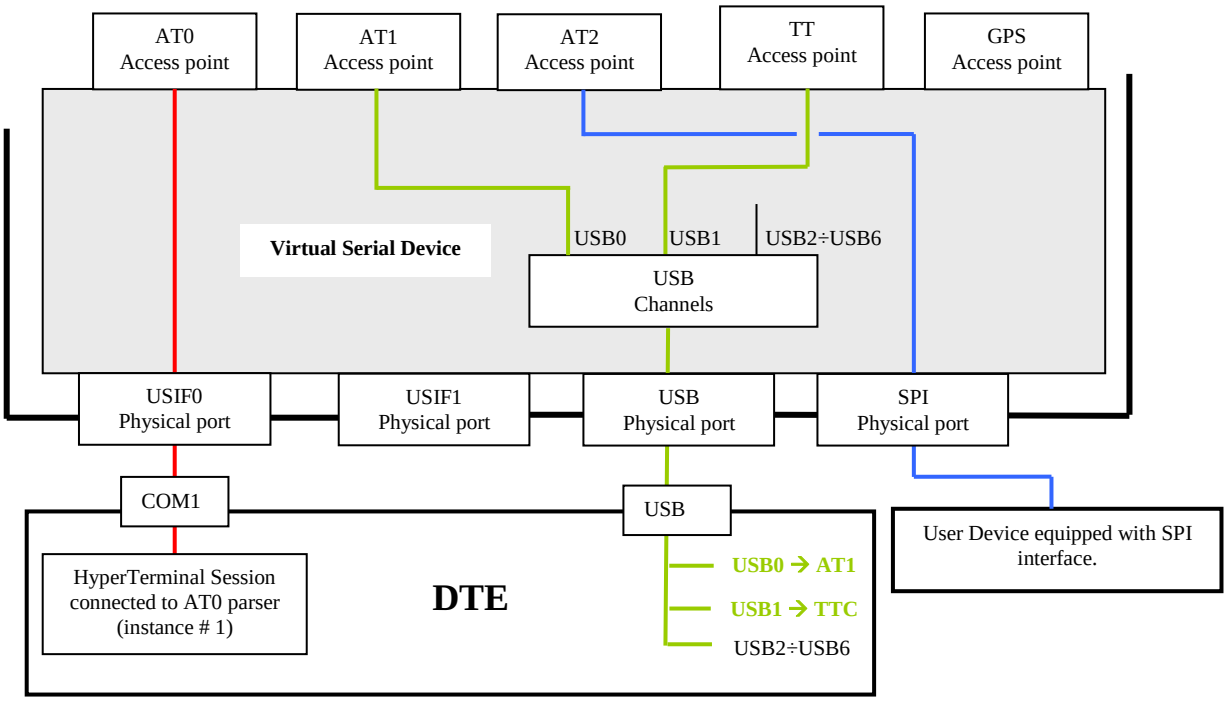

Fig. 6: #PORTCFG=2 + USB Cable

4.4. AT#PORTCFG=3

AT#PORTCFG=3					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1			X		Ref. chapter 6.1.1
SPI					

Tab. 11: #PORTCFG=3, no USB Cable

AT#PORTCFG=3					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2					
USB3					
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1			X		Ref. chapter 6.1.1
SPI					

Tab. 12: #PORTCFG=3, with USB Cable

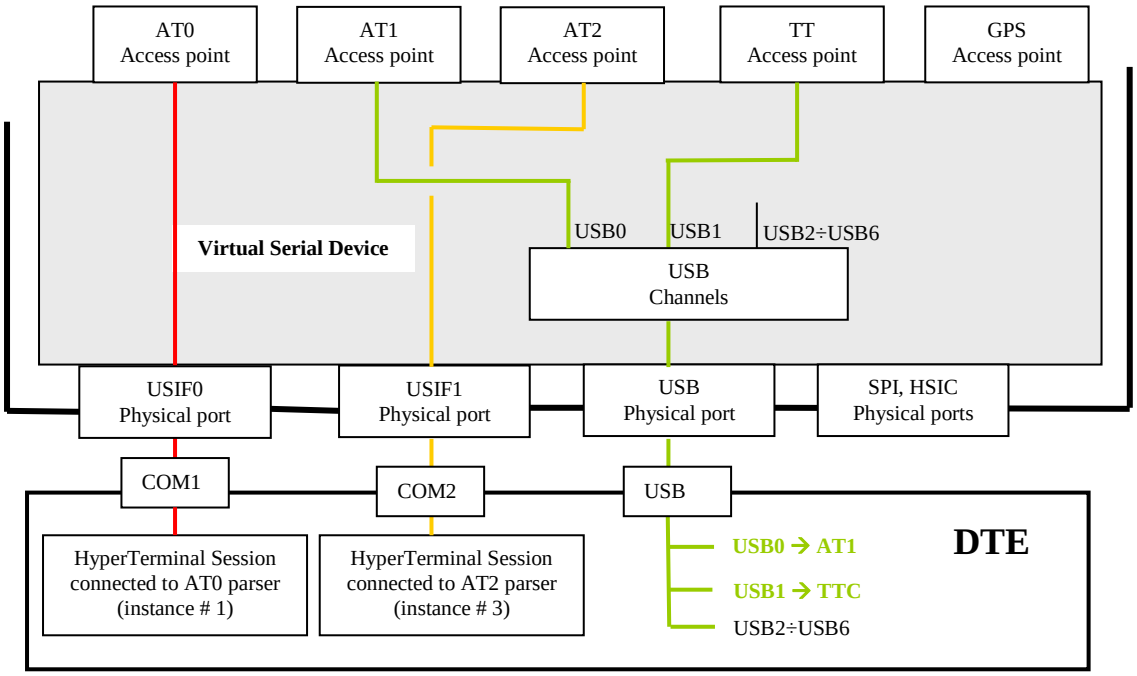

Fig. 7: #PORTCFG=3 + USB Cable

4.5. AT#PORTCFG=4

AT#PORTCFG=4					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0		X			Ref. chapter 6.1.1.4
USIF1					
SPI			X		Ref. chapter 6.1.1.4

Tab. 13: #PORTCFG=4, no USB Cable

AT#PORTCFG=4					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0	X				Ref. chapter 6.1.1
USB1				TTC	
USB2					
USB3			X		Ref. chapter 6.1.1
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0		X			Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 14: #PORTCFG=4, with USB Cable

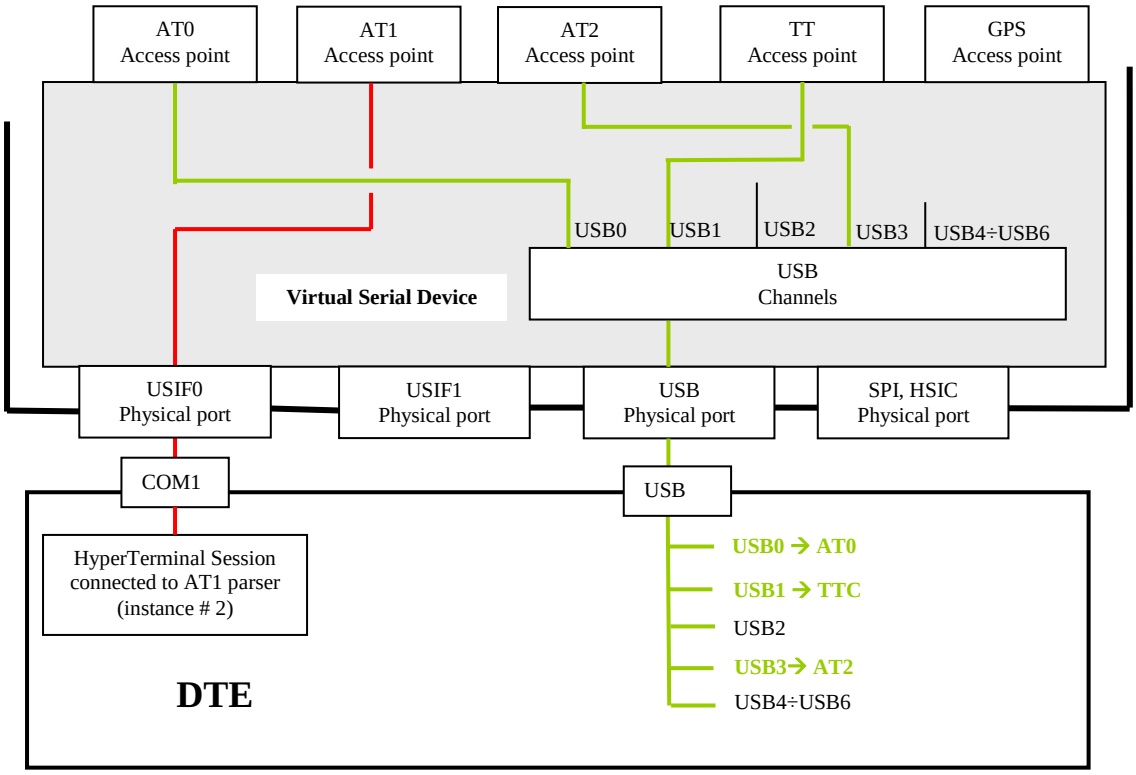

Fig. 8: #PORTCFG=4 + USB Cable

4.6. AT#PORTCFG=5

AT#PORTCFG=5					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI			X		Ref. chapter 6.1.1

Tab. 15: #PORTCFG=5, no USB Cable

AT#PORTCFG=5					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2					
USB3	X				Ref. chapter 6.1.1
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI			X		Ref. chapter 6.1.1

Tab. 16: #PORTCFG=5, with USB Cable

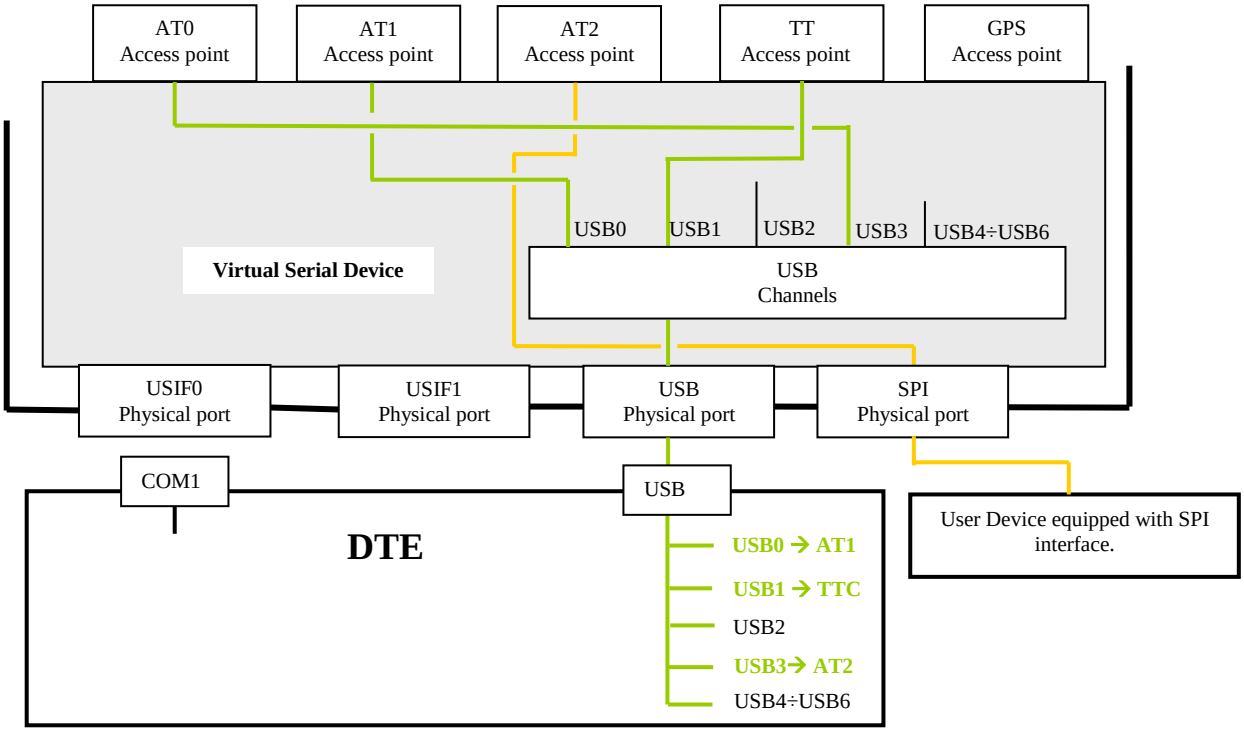

Fig. 9: #PORTCFG=5 + USB Cable

4.7. AT#PORTCFG=6

AT#PORTCFG=6					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0			X		Ref. chapter 6.1.1
USIF1					
SPI	X				Ref. chapter 6.1.1

Tab. 17: #PORTCFG=6, no USB Cable

AT#PORTCFG=6					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2					
USB3					
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0			X		Ref. chapter 6.1.1
USIF1					
SPI	X				Ref. chapter 6.1.1

Tab. 18: #PORTCFG=6, with USB Cable

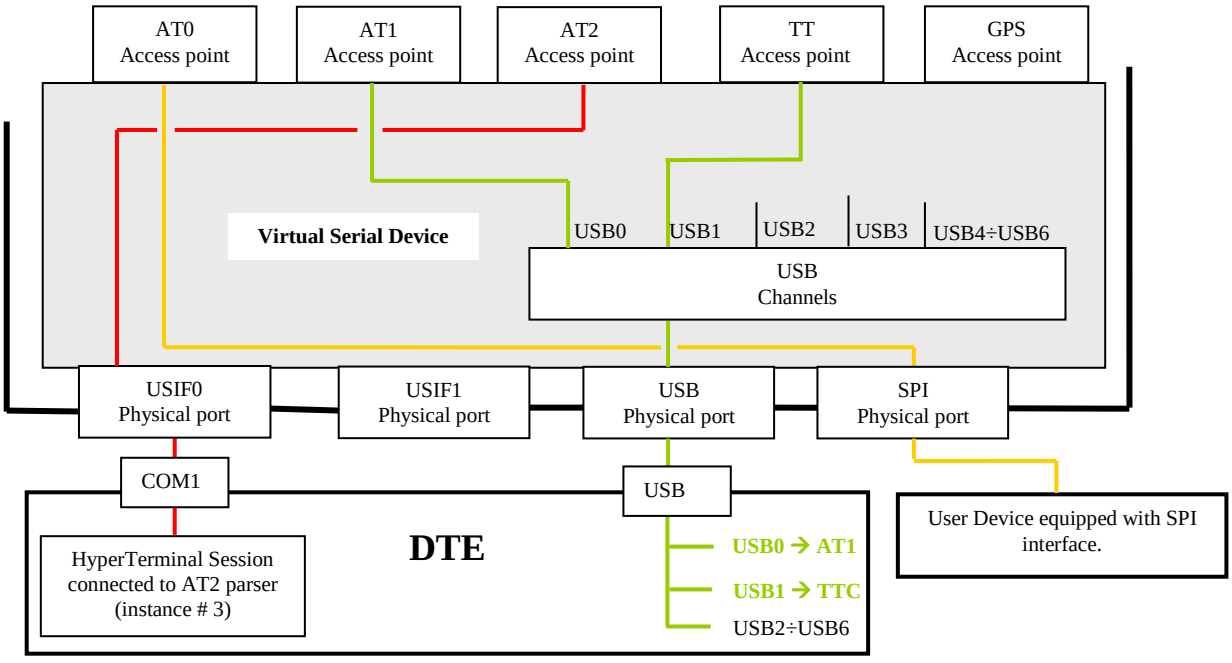

Fig. 10: #PORTCFG=6 + USB Cable

4.8. AT#PORTCFG=7

AT#PORTCFG=7					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 19: #PORTCFG=7, no USB Cable

AT#PORTCFG=7					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2				3G	
USB3			X		Ref. chapter 6.1.1
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 20: #PORTCFG=7, with USB Cable

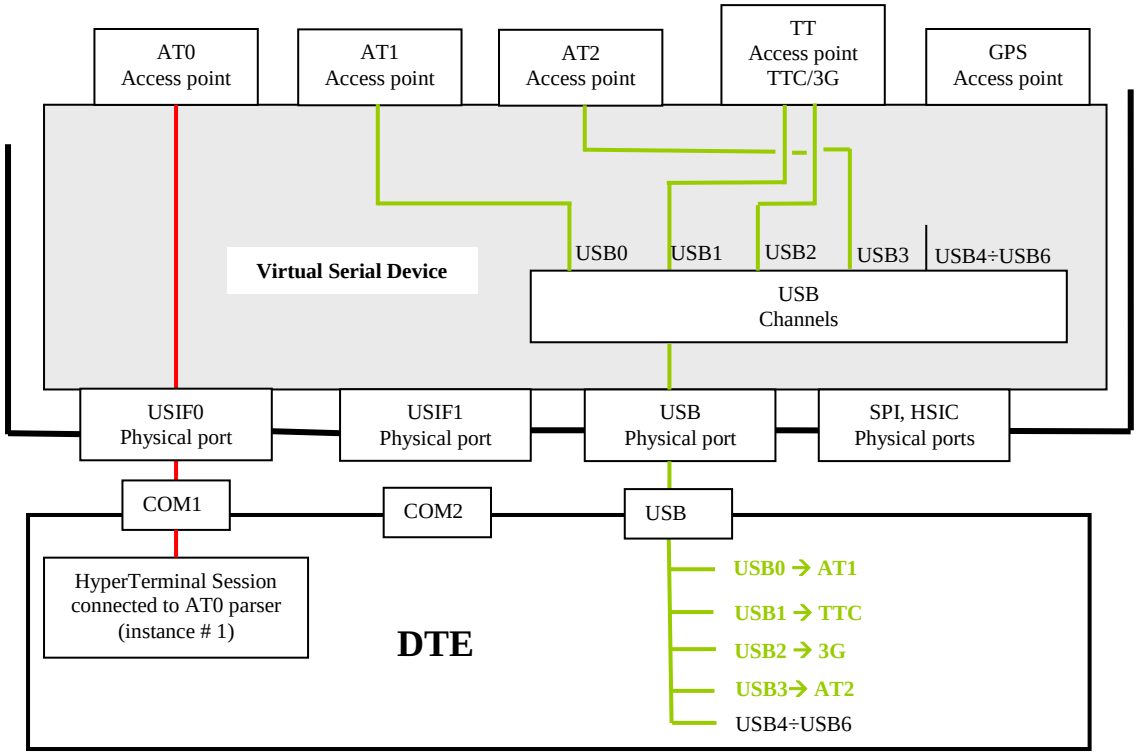

Fig. 11: #PORTCFG=7 + USB Cable

4.9. AT#PORTCFG=8

AT#PORTCFG=8					
	AT0	AT1	AT2	TT	GPS/NMEA
NO USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI					

Tab. 21: #PORTCFG=8, no USB Cable

AT#PORTCFG=8					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0	X				
USB1				TTC	
USB2					
USB3		X			
USB4			X		
USB5					Ref. chapter 6.1.1.5
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI					

Tab. 22: #PORTCFG=8, with USB Cable

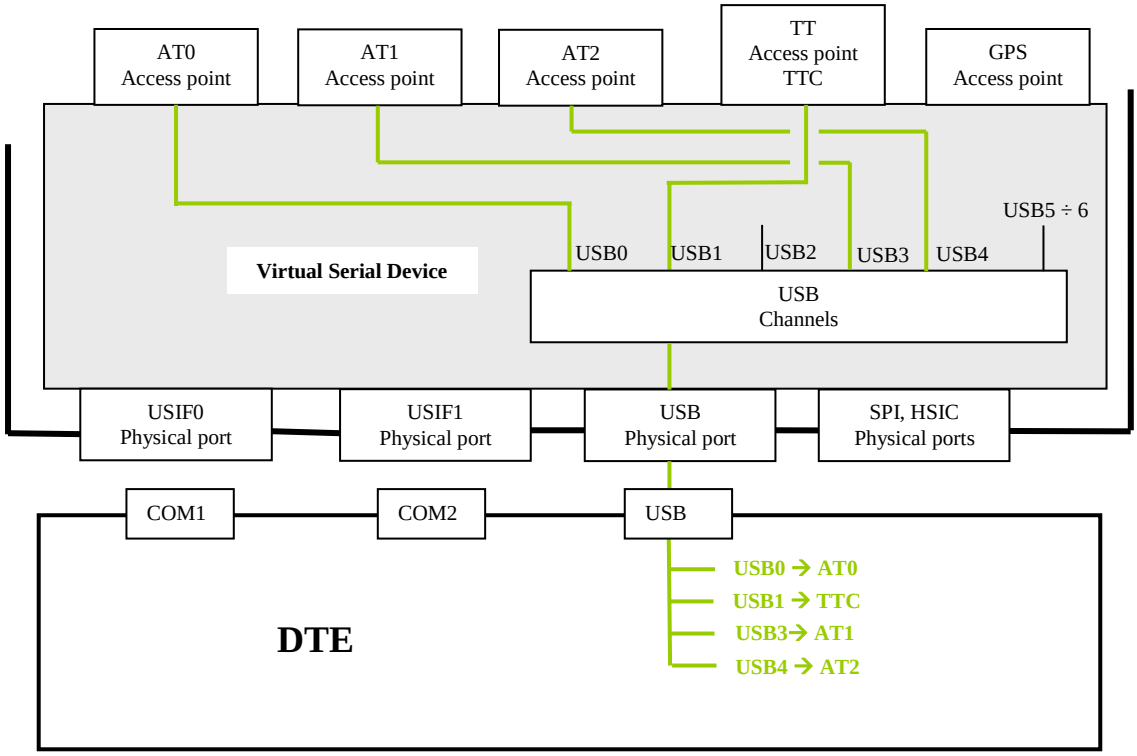

Fig. 12: #PORTCFG=8 USB Cable Only

4.10. AT#PORTCFG=9

AT#PORTCFG=9					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0			X		Ref. chapter 6.1.1
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 23: #PORTCFG=9, no USB Cable

AT#PORTCFG=9					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			Ref. chapter 6.1.1
USB1				TTC	
USB2				3G	
USB3					
USB4					
USB5					
USB6					
USBHSI0			X		Ref. chapter 6.1.1
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 24: #PORTCFG=9, with USB Cable

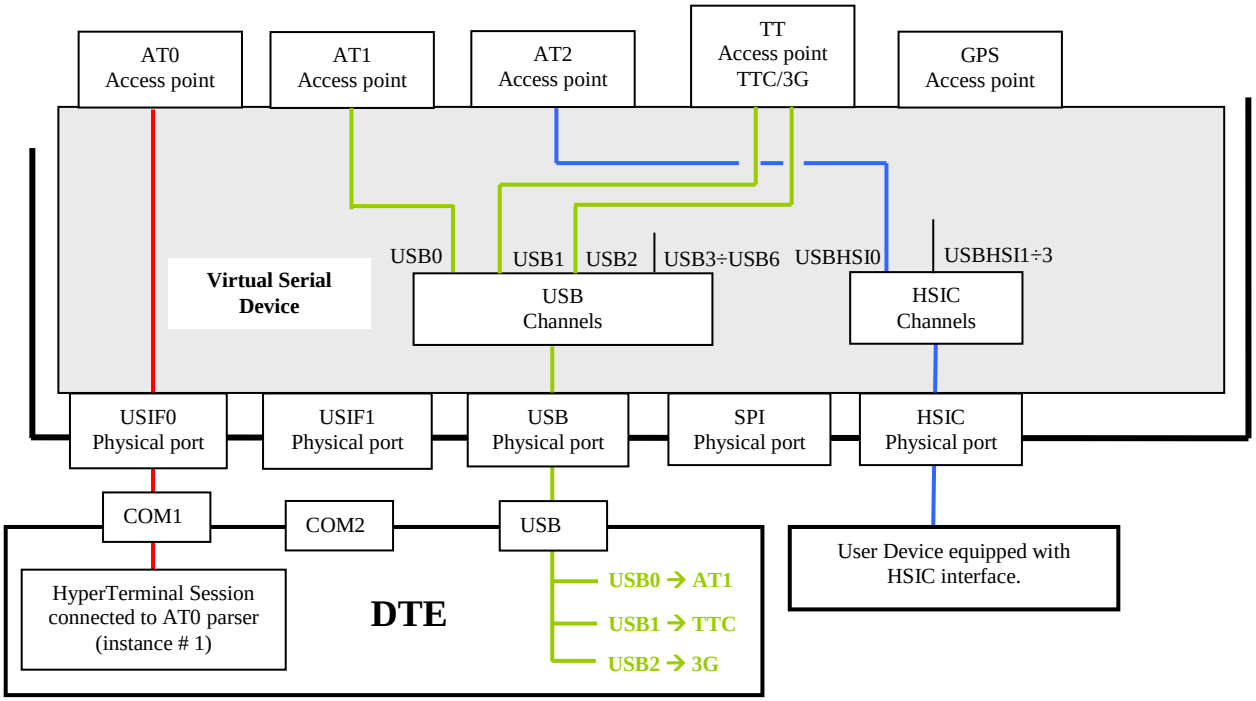

Fig. 13: #PORTCFG=9 + USB Cable

4.11. AT#PORTCFG=10

AT#PORTCFG=10					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0	X				Ref. chapter 6.1.1
USBHSI1		X			Ref. chapter 6.1.1
USBHSI2					
USBHSI3					
USIF0			X		Ref. chapter 6.1.1
USIF1					
SPI					

Tab. 25: #PORTCFG=10, no USB Cable

AT#PORTCFG=10					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0			X		Ref. chapter 6.1.1
USB1				TTC	
USB2				3G	
USB3					
USB4					
USB5					
USB6					
USBHSI0	X				Ref. chapter 6.1.1
USBHSI1		X			Ref. chapter 6.1.1
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI					

Tab. 26: #PORTCFG=10, with USB Cable

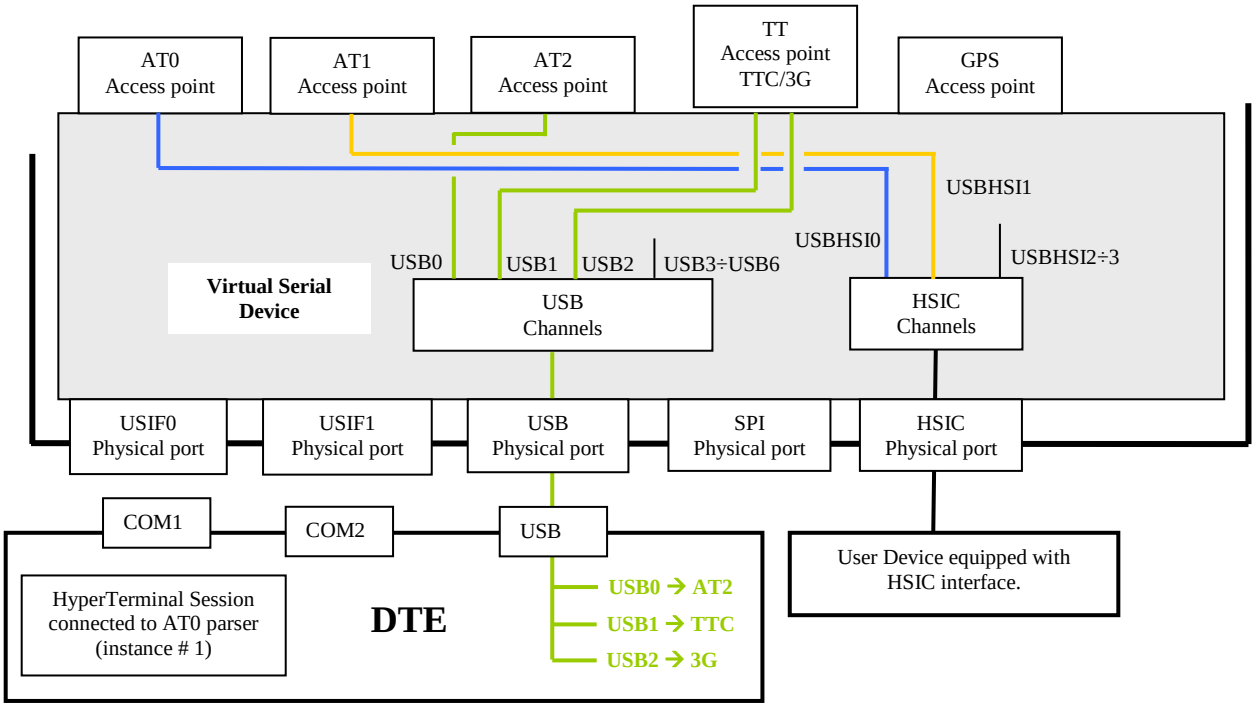

Fig. 14: #PORTCFG=10 + USB Cable

4.12. AT#PORTCFG=11

NOTICE: use this command in conjunction with an external GPS receiver connected to the module through USIF1 serial port; see Applicability Table to know the modules supporting this command.

AT#PORTCFG=11					
	AT0	AT1	AT2	TT	Ext. GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.2.1
USIF1					Ext. GPS Receiver connected to USIF1
SPI					

Tab. 27: #PORTCFG=11, no USB Cable

AT#PORTCFG=11					
	AT0	AT1	AT2	TT	Ext. GPS/NMEA
USB0		X			Ref. chapter 6.1.2.2
USB1				TTC	
USB2					
USB3			X		Ref. chapter 6.1.2.2
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				Ref. chapter 6.1.2.2
USIF1					Ext. GPS Receiver connected to USIF1
SPI					

Tab. 28: #PORTCFG=11, with USB Cable

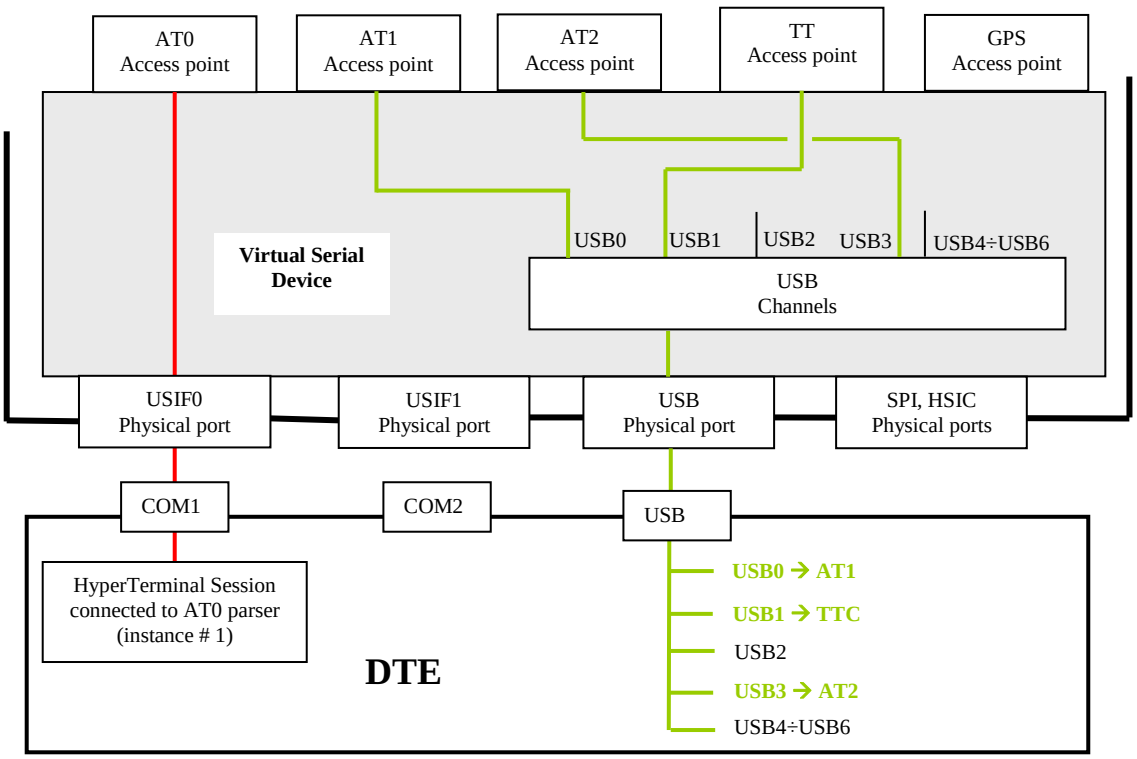

Fig. 15: #PORTCFG=11 + USB Cable

4.13. AT#PORTCFG=12

AT#PORTCFG=12 provides two new access points: AT3, DLink.

AT#PORTCFG=12							
	AT0	AT1	AT2	AT3	TT	Ext. GPS	DLink
No USB cable							
USBHSI0							
USBHSI1							
USBHSI2							
USBHSI3							
USIF0	X						
USIF1							
SPI							

Tab. 29: #PORTCFG=12, no USB Cable

AT#PORTCFG=12							
	AT0	AT1	AT2	AT3	TT	Ext. GPS	DLink
USB0		X					
USB1					TT		
USB2							
USB3			X				
USB4				X			
USB5							Ref. chapter 5.3
USB6							
USBHSI0							
USBHSI1							
USBHSI2							
USBHSI3							
USIF0	X						
USIF1							
SPI							

Tab. 30: #PORTCFG=12, with USB Cable

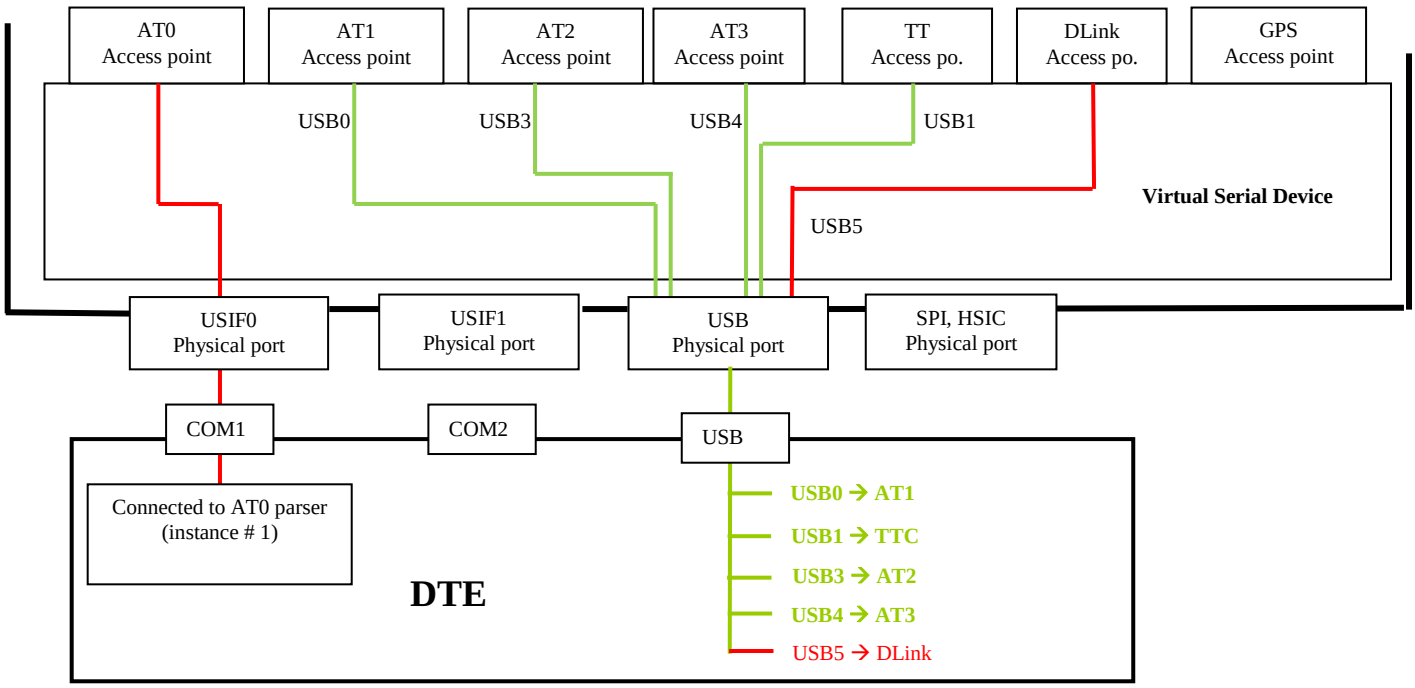

Fig. 16: #PORTCFG=12 + USB Cable

5. CMUX Protocol

This section shows examples of ports arrangement using CMUX protocol. If you need to develop a Multiplexing Protocol running on your application processor (e.g. a user micro-controller), refer to document [1] to get detailed information.

5.1. CMUX Protocol on USIF0 Serial Port

Here is an example of ports arrangement supporting CMUX protocol on USIF0 serial port.

Assume that the module is configured as indicated in Fig. 2: #PORTCFG=0, and no USB cable plugged in. In addition, suppose that the used DTE is a Windows-PC, and Fig. 17 shows its device configuration. Now, run on the DTE the TELIT Serial Port MUX application configured as shown in Fig. 18, and connect the MUX application to COM1 physical port, refer to Fig. 19. When the user starts an application (e.g. Hyper Terminal) connected to one of the three Virtual Ports (VCOM20 ÷ VCOM22), TELIT Serial Port MUX application sends automatically the AT+CMUX=0 command to the module and the CMUX protocol is activated.

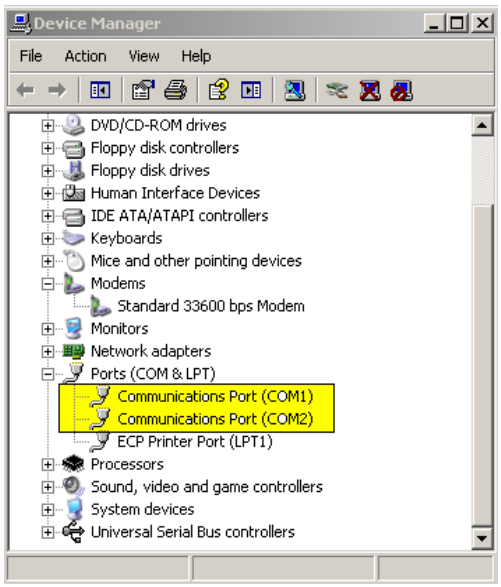

Fig. 17: Physical COMx Ports

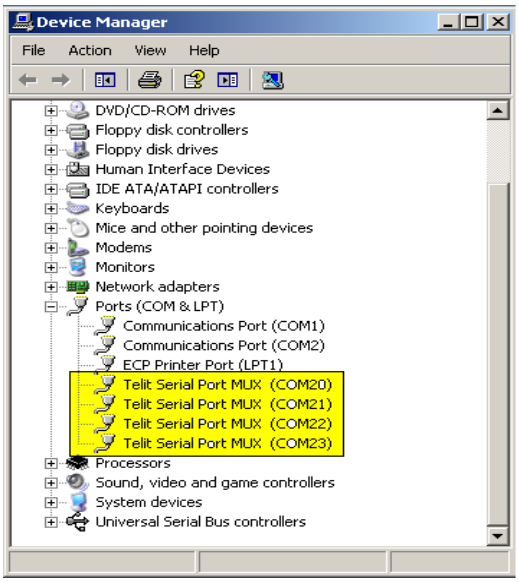

Fig. 18: Virtual Serial Ports of MUX

The configuration of the TELIT Serial Port MUX application must avoid virtual serial ports conflict with the physical or virtual serial ports already present on the Windows-PC. Tab. 31 summarizes the new configuration.

Module ←→DTE connection	VCOMx → VCx	AT0	AT1	AT2	TT	GPS/NMEA
USB not used						
HSIC not used						
USIF0 ←→COM1	VCOM20→VC1	X				
	VCOM21→VC2		X			
	VCOM22→VC3			X		
	VCOM23→VC4					
USIF1 not used						
SPI not used						

Tab. 31: Ports Arrangement with CMUX Connected to USIF0

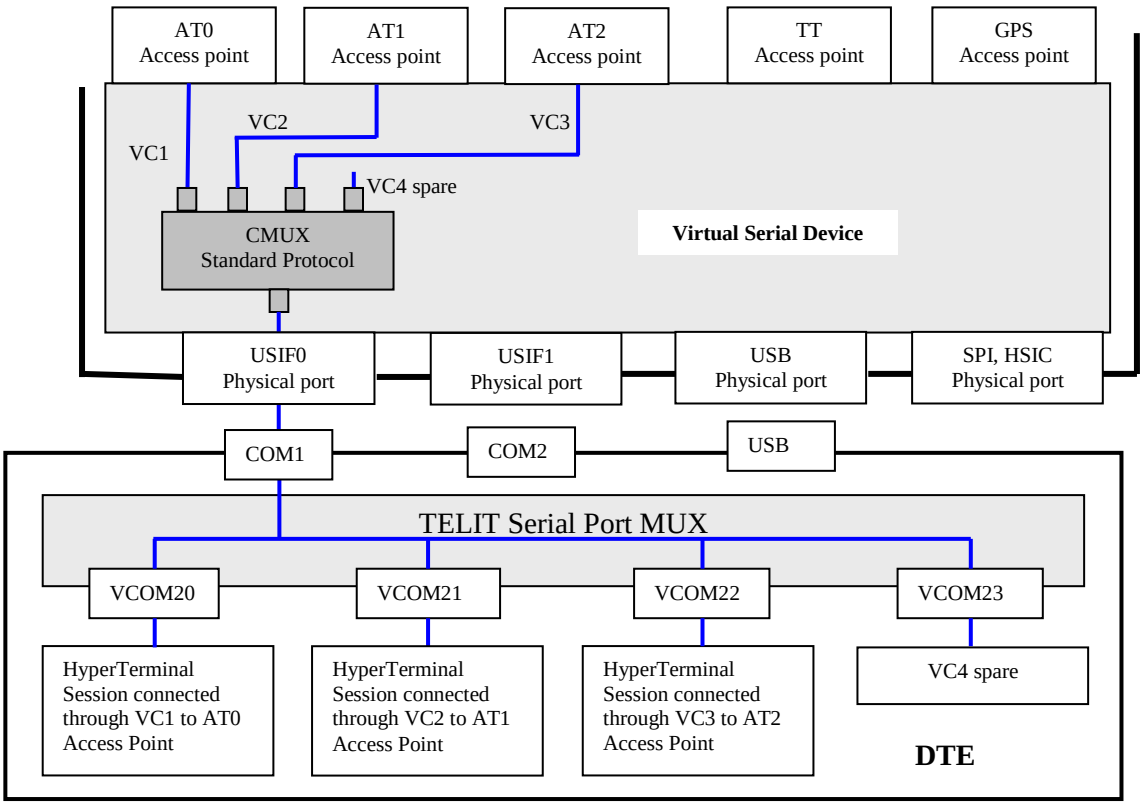

Fig. 19: CMUX Connected to USIF0

5.1.1. Connection with TTC Application

If TTC connection is needed, start from the configuration: #PORTCFG=1, no USB cable, see Tab. 7. Follow the steps stated above, and refer to Fig. 20. Tab. 32 summarizes the new configuration.

Module ←→DTE connection	VCOMx → VCx	AT0	AT1	AT2	TT	GPS/NMEA
USB not used						
HSIC not used						
USIF0 ←→ COM1	VCOM20→VC1	X				
	VCOM21→VC2		X			
	VCOM22→VC3			X		
	VCOM23→VC4					
USIF1 ←→ COM2					TTC	
SPI not used						

Tab. 32: Ports Arrangement with CMUX + TTC

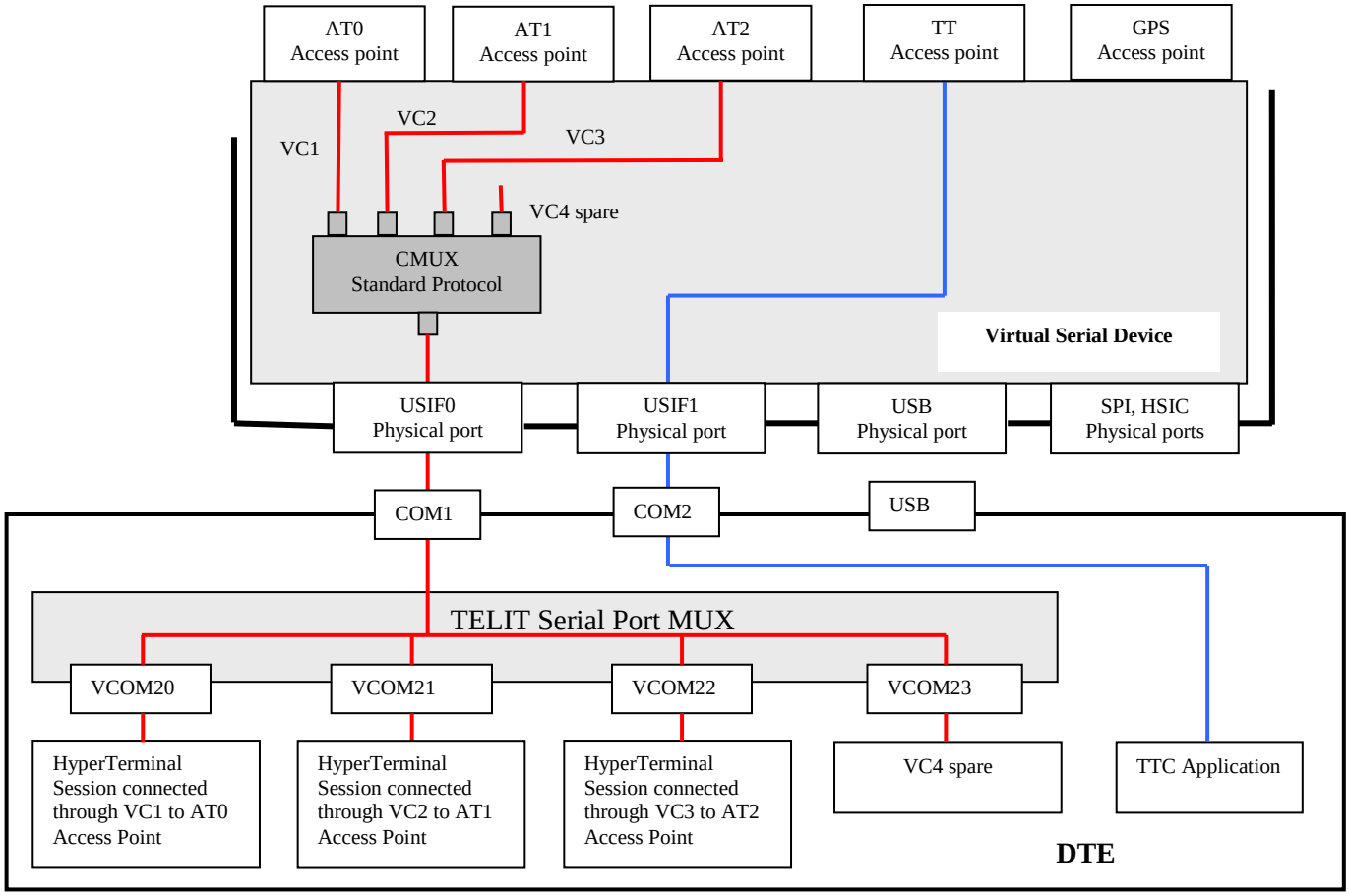

Fig. 20: CMUX Connected to USIF0 + TTC Connected to USIF1

5.2. CMUX Protocol on USB Channel

Here is an example of ports arrangement supporting CMUX protocol on USB channel

Assume that the module is configured as indicated in Fig. 4: #PORTCFG=0, and USB cable plugged in. In addition, suppose that the used DTE is a Windows PC, and Fig. 3 shows its device configuration. Now, run on the DTE the TELIT Serial Port MUX application configured as shown in Fig. 21, and connect the MUX application to USB3 channel mapped into VCOM8 virtual port as shown on Tab. 4, refer to Fig. 22. When the user starts an application (e.g. Hyper Terminal) connected to one of the three Virtual Ports (VCOM20 ÷ VCOM22), TELIT Serial Port MUX application sends automatically the AT+CMUX=0 command to the module and the CMUX protocol is activated.

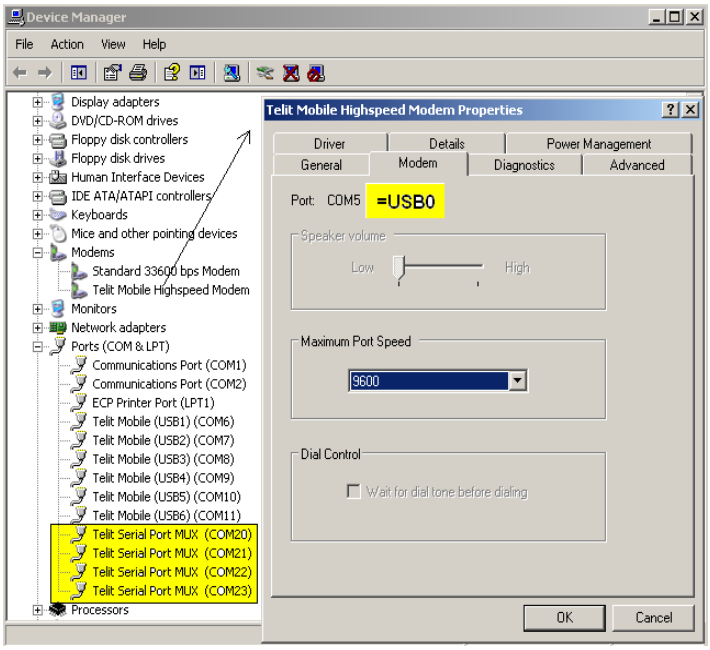

Fig. 21: Virtual Serial Ports of Telit Serial Port MUX

Tab. 33 summarizes the new configuration.

Module ←→DTE connection	Channels	USBx → VCOM	VCOMx → VCx	AT0	AT1	AT2	TT	GPS/NMEA
USB ←→ USB	USB0							
	USB1						TTC	
	USB2							
	USB3	VCOM8	VCOM20 → VC1	X				Ref. chapter 6.1.1.3
			VCOM21 → VC2		X			Ref. chapter 6.1.1.3
			VCOM22 → VC3			X		Ref. chapter 6.1.1.3
			VCOM23 → VC4					
	USB4							
	USB5							
	USB6							
HSIC not used								
USIF0 not used								
USIF1 not used								
SPI not used								

Tab. 33: Ports Arrangement with CMUX Connected to USB3 Channel

NOTICE: AT0 (instance # 1) is disconnected from USIF0 and connected to USB3/VCOM8/VCOM20→VC1, the TTC stays on USB1.

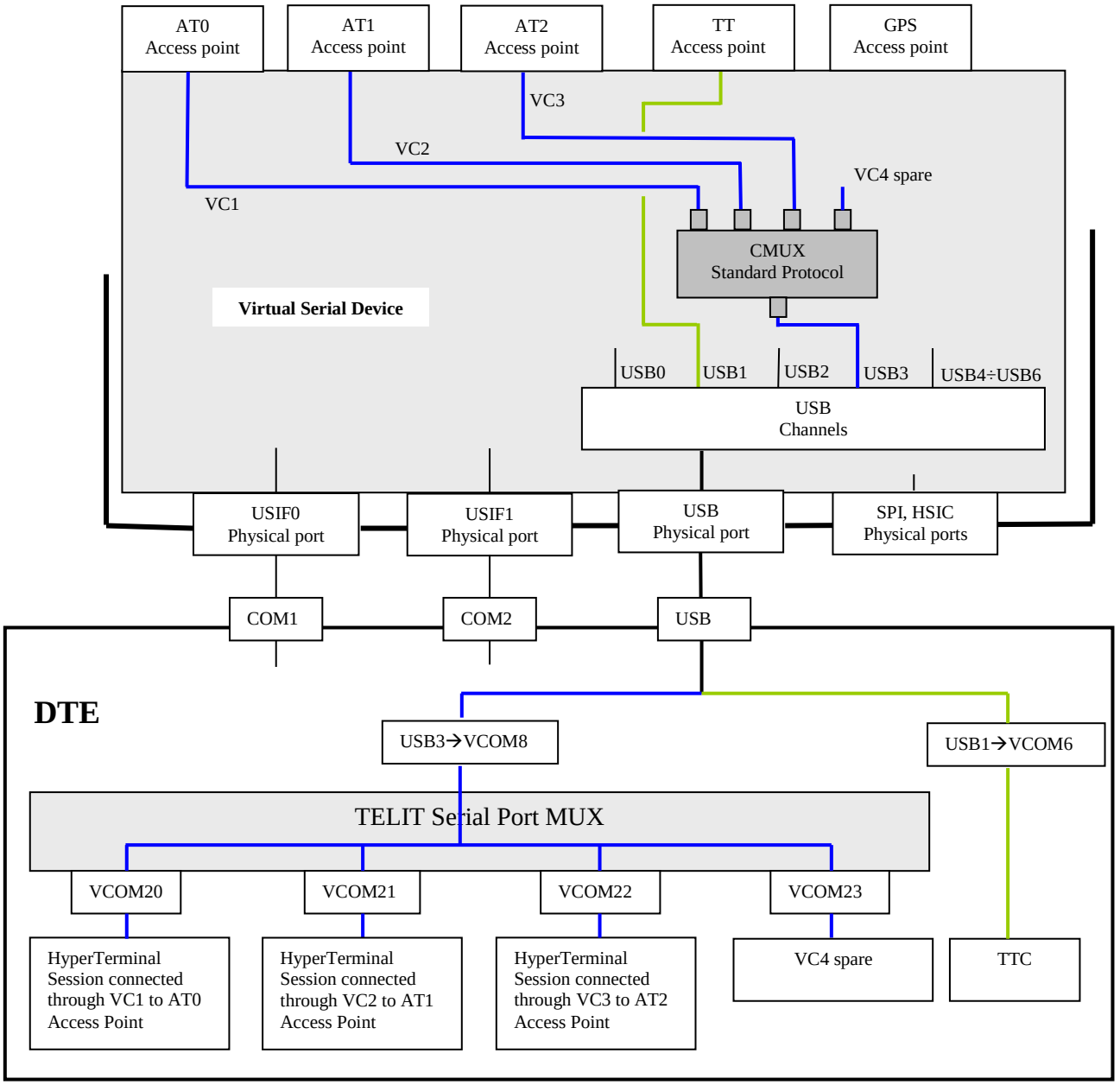

Fig. 22: CMUX Connected to USB3 Channel

5.3. CMUX Protocol and AT#PORTCFG=12

The module provide the DLink access point that permits the logical connection between two user applications (named for example: "A" and "B") running on two different devices connected to the module by means of serial lines. Assume that the module is connected to the following devices, refer to Fig. 23:

Window-PC (DTE_A) equipped with the Telit Serial Port MUX tool is running:

- user application "A"
- user application to send AT commands and receive URC (e.g. Hyper Terminal)

Window-PC (DTE_B) is running:

- user application "B"
- user PPP application
- user application to send AT commands and receive URC (e.g. Hyper Terminal)

The module is configured as indicated in Fig. 16: #PORTCFG=12, and USB cable plugged in. The used DTE is a Windows-PC, and Fig. 17 shows its device configuration. Run on the DTE the TELIT Serial Port MUX application configured as shown in Fig. 18, and connect the MUX application to COM1 physical port, refer to Fig. 23. When the user starts an application (e.g. Hyper Terminal) connected to one of the two Virtual Ports (VCOM20, VCOM21), TELIT Serial Port MUX application sends automatically the AT+CMUX=0 command to the module and the CMUX protocol is activated.

Use the AT#DLINK command to enable the logical internal connection between AT0 and DLink access points. Now, user application "A" is connected to user application "B" as shown in Fig. 23. Refer to document [2] to have information on the AT#DLINK syntax.

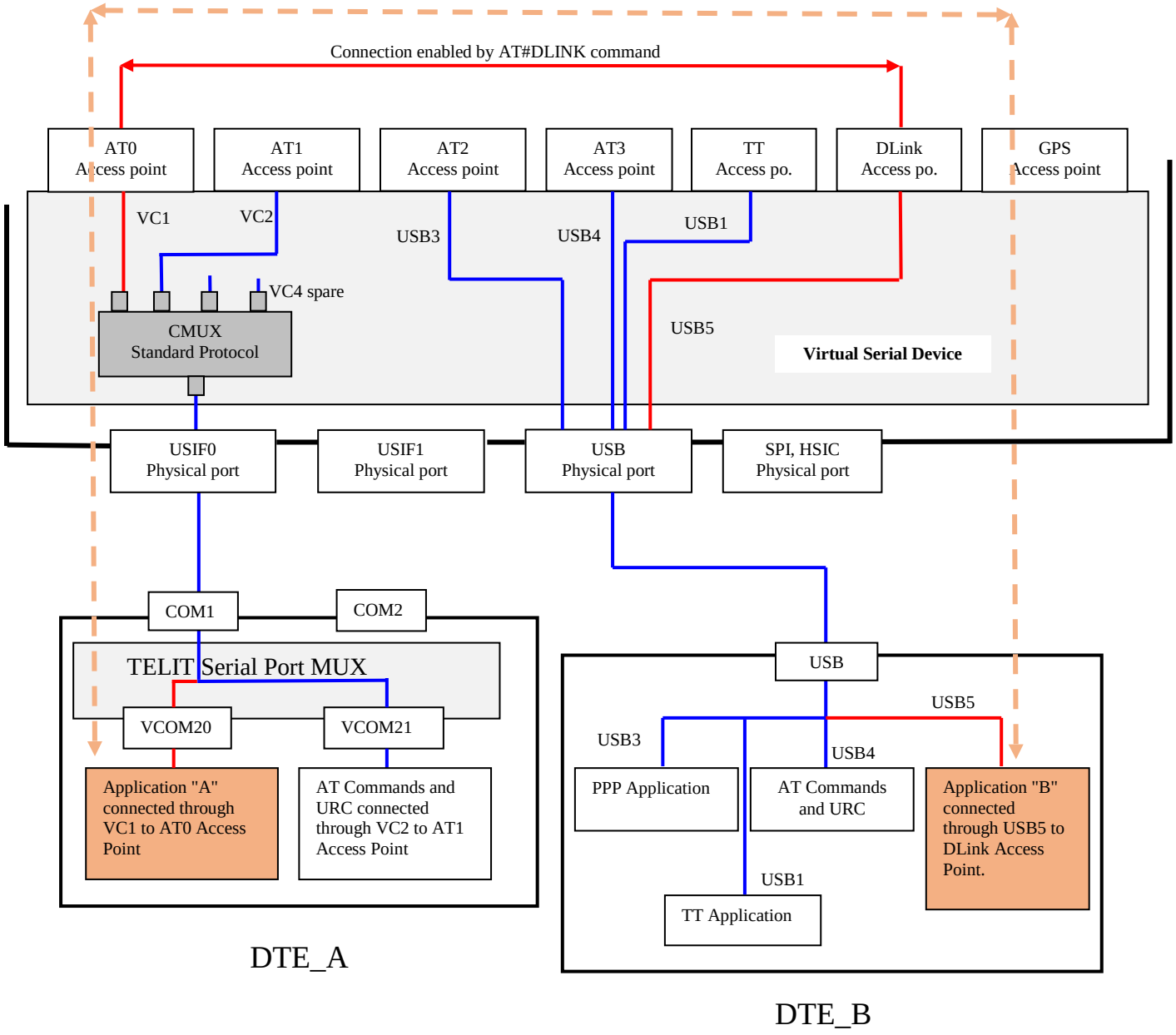

Fig. 23: CMUX & AT#PORTCFG=12

6. Services

The modules families covered by the present document provide the services as indicated in the Applicability Table.

Different Access points connect these services to the Virtual Serial Device software layer. This section describes how the user can access the supported service by means of the external physical serial ports, which in their turn are connected to the Virtual Serial Device.

6.1. GPS

In general, the GPS receiver can be “built-in” or external to the module.

6.1.1. GPS/NMEA Sentences from Built-in GPS

The built-in GPS receiver can send NMEA sentences on different physical ports, in accordance with the current ports configuration. In general, NMEA sentences run on the physical port used by the operator to enter the AT$GPSP and AT$GPSNMUN commands; in this case AT commands and NMEA sentences share the same physical port at the same time. Refer to document [2] to have information on AT commands syntax.

The following sub-chapters show some detailed examples of logical connections settings.

6.1.1.1. AT#PORTCFG=0

Tab. 5 summarizes the starting configuration of the module (#PORTCFG=0). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered through USIF0 port. AT0 parser executes the AT commands, and after that NMEA sentences and AT commands run on USIF0 port as summarized in Tab. 34. See also Fig. 24.

AT#PORTCFG=0					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				X
USIF1					
SPI					

Tab. 34: USIF0 port supports NMEA sentences

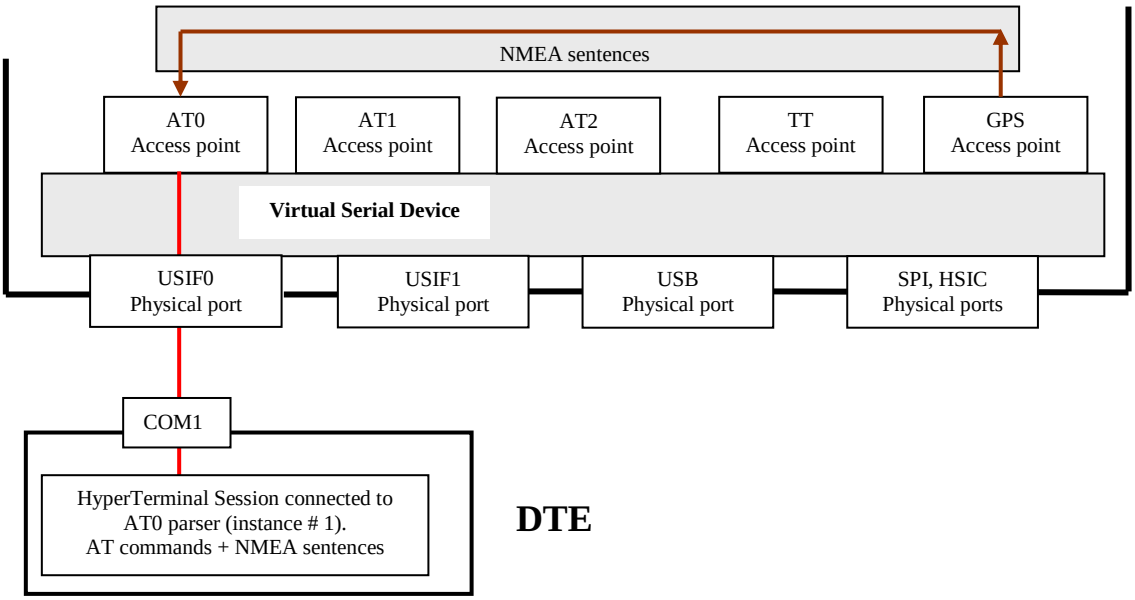

Fig. 24: USIF0 Port Supports AT Commands + NMEA Sentences

6.1.1.2. AT#PORTCFG=0 + USB

Tab. 6 summarizes the starting configuration of the module (#PORTCFG=0+USB). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered, for example, through USB0 channel. AT1 parser executes the AT commands, and after that NMEA sentences and AT commands run on USB0 channel as summarized in Tab. 35. See also Fig. 25.

AT#PORTCFG=0					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0		X			X
USB1				TTC	
USB2					
USB3			X		
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				
USIF1					
SPI					

Tab. 35: USB0 Channel Supports NMEA Sentences

NOTICE: the user can issue the AT commands through USIF0/AT0, USB0/AT1, or USB3/AT2 channel/parser. The NMEA sentences are routed respectively on USIF0, USB0 (as shown by the example), or USB3 channel.

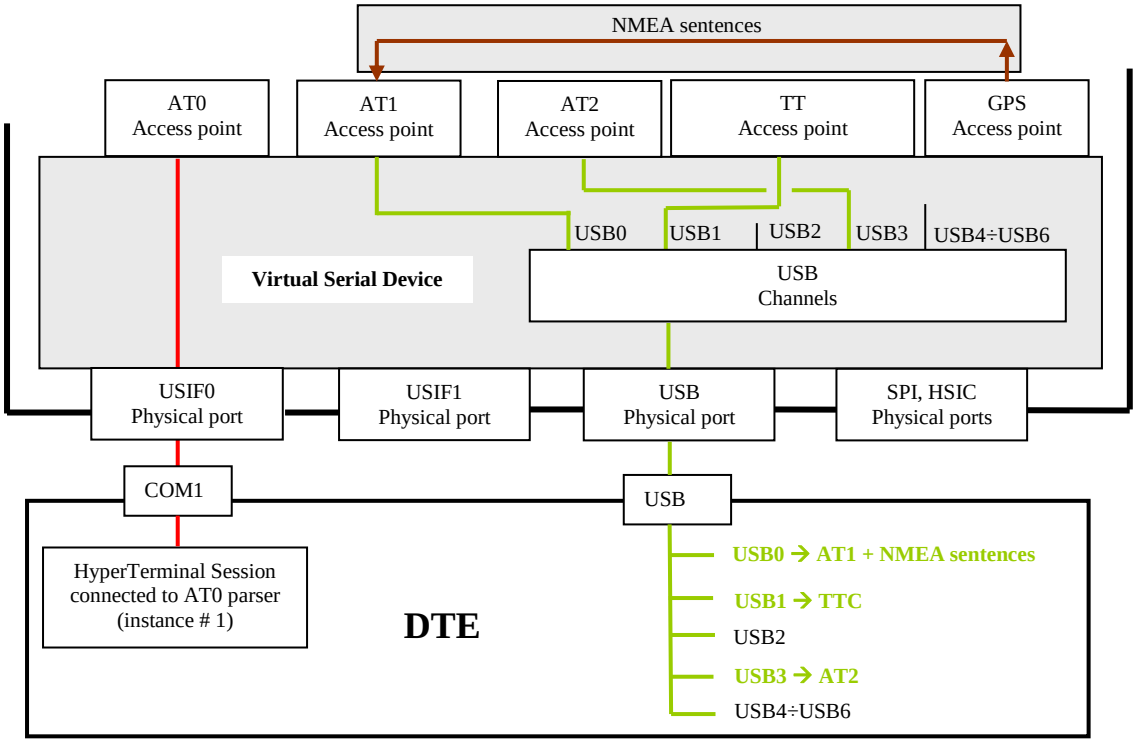

Fig. 25: USB0 Channel Supports AT Commands + NMEA Sentences

6.1.1.3. AT#PORTCFG=0 + USB + CMUX

Here is an example of ports arrangement running NMEA sentences on USB channel using CMUX protocol.

Tab. 33 summarizes the starting configuration of the module (#PORTCFG=0+USB+CMUX). Now, enable GPS/NMEA sentences via the AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered, for example, through USB3-VC3 port. AT2 parser executes the AT commands, and after that NMEA sentences and AT commands run on VCOM22→VC3 channel as summarized in Tab. 36. See also Fig. 26.

Module←→DTE connection	Channels	USBx → VCOM	VCOMx → VCx	AT0	AT1	AT2	TT	GPS/NMEA
USB ←→ USB	USB0							
	USB1						TTC	
	USB2							
	USB3	VCOM8	VCOM20 → VC1	X				
			VCOM21 → VC2		X			
			VCOM22 → VC3			X		X
			VCOM23 → VC4					
	USB4							
	USB5							
	USB6							
HSIC not used								
USIF0 not used								
USIF1 not used								
SPI not used								

Tab. 36: USB3-VC3 Channel Supports AT Commands + NMEA Sentences

NOTICE: the user can issue the AT commands through VCOM20 → VC1/AT0, VCOM21 → VC2/AT1, or VCOM22 → VC3/AT2 channel/parser. The NMEA sentences are routed respectively on VC1, VC2, or VC3 channel. The example works with VC3 channel.

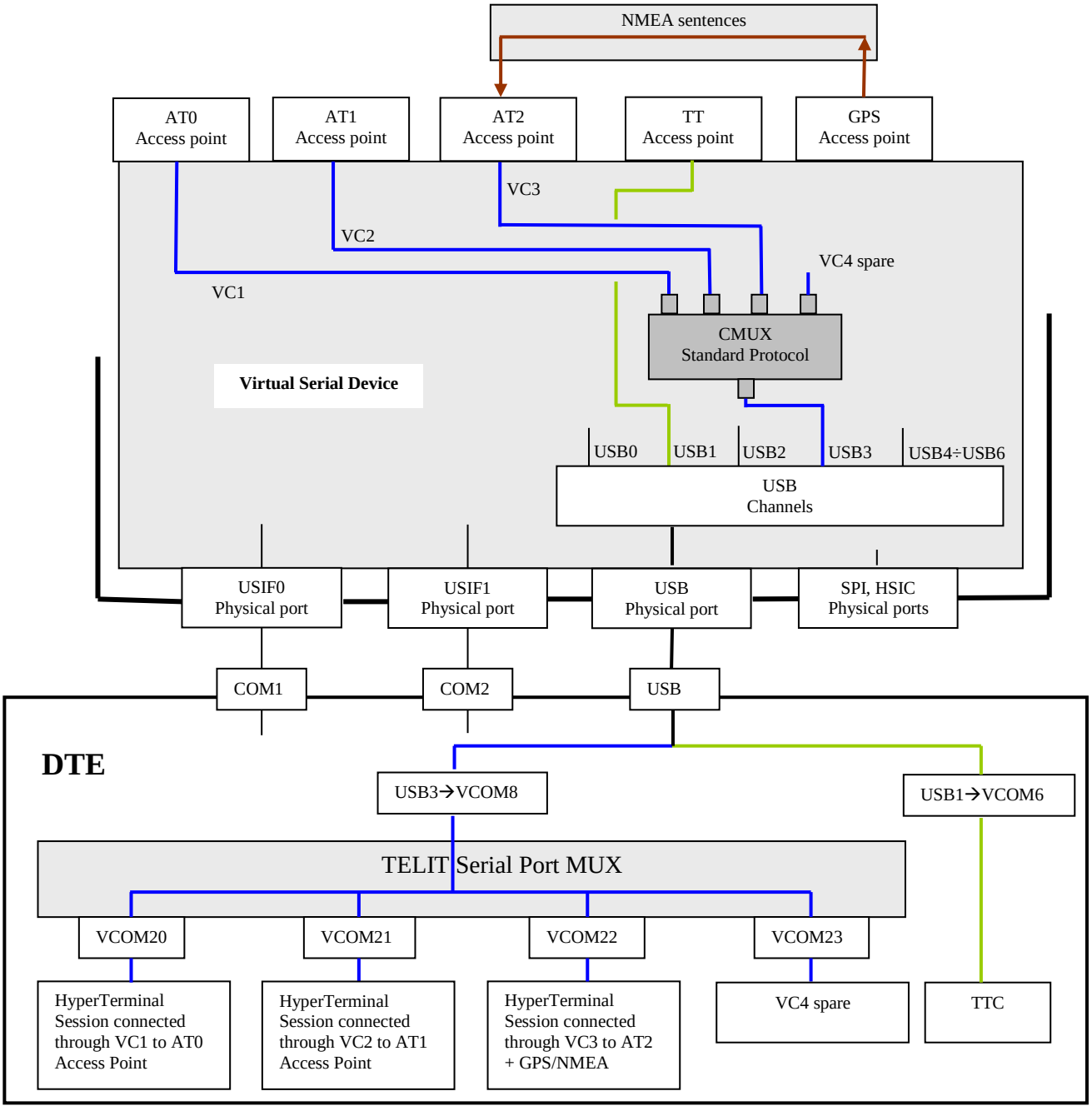

Fig. 26: USB3-VC3 Channel Supports AT Commands + NMEA Sentences

6.1.1.4. AT#PORTCFG=4

Let us assume that Tab. 13 summarizes the starting configuration of the module (#PORTCFG=4). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered, for example, through SPI port. AT2 parser executes the AT commands, and after that NMEA sentences and AT commands run on SPI port as summarized in Tab. 37. See also the Fig. 27.

AT#PORTCFG=4					
	AT0	AT1	AT2	TT	GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0		X			
USIF1					
SPI			X		X

NOTICE: the user can issue the AT commands through USIF0/AT1, or SPI/AT2 port/parser. The NMEA sentences are routed respectively on USIF0 or SPI port, as shown by the example.

Tab. 37: SPI Port Supports NMEA Sentences

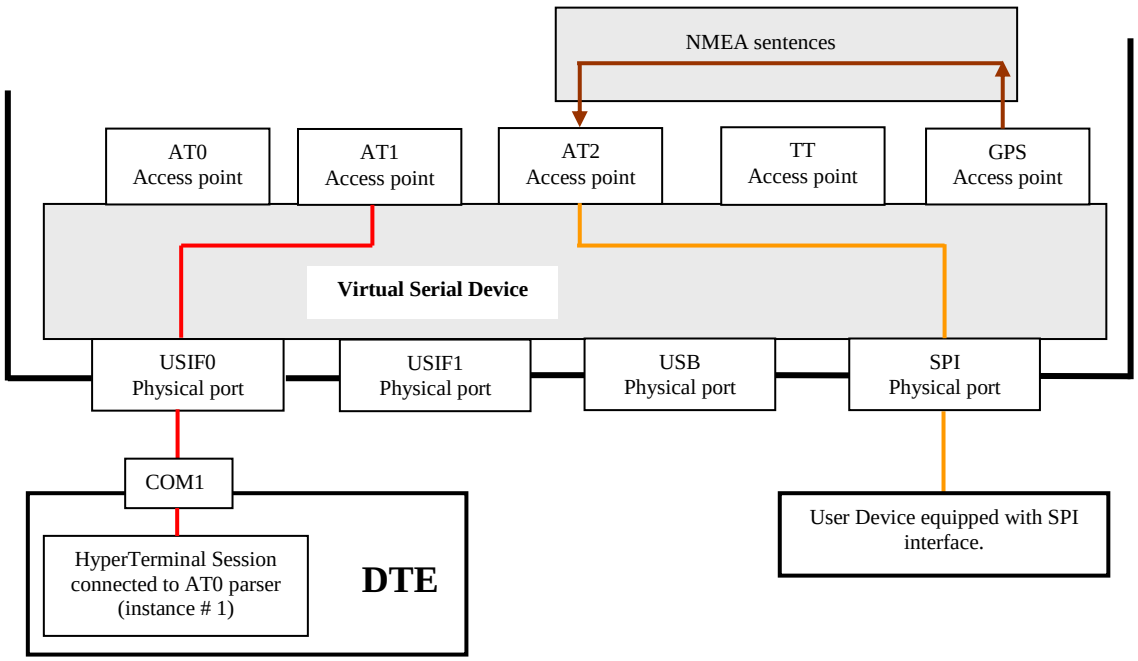

Fig. 27: SPI Port Supports AT Commands + NMEA Sentences

6.1.1.5. AT#PORTCFG=8

Tab. 22 summarizes the starting configuration of the module (#PORTCFG=8). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered, for example, through USB3 channel. AT1 parser executes the AT commands, and after that NMEA sentences run on USB5 channel as summarized in Tab. 38. See also Fig. 28.

AT#PORTCFG=8					
	AT0	AT1	AT2	TT	GPS/NMEA
USB0	X				
USB1				TTC	
USB2					
USB3		X			
USB4			X		
USB5					X
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0					
USIF1					
SPI					

Tab. 38: USB3 Channel Supports NMEA Sentences

NOTICE: the user can issue the AT commands through USB0/AT0, USB3/AT1, and USB4/AT2 channels/parsers. In any case, the GPS/NMEA sentences are routed on USB5 channel.

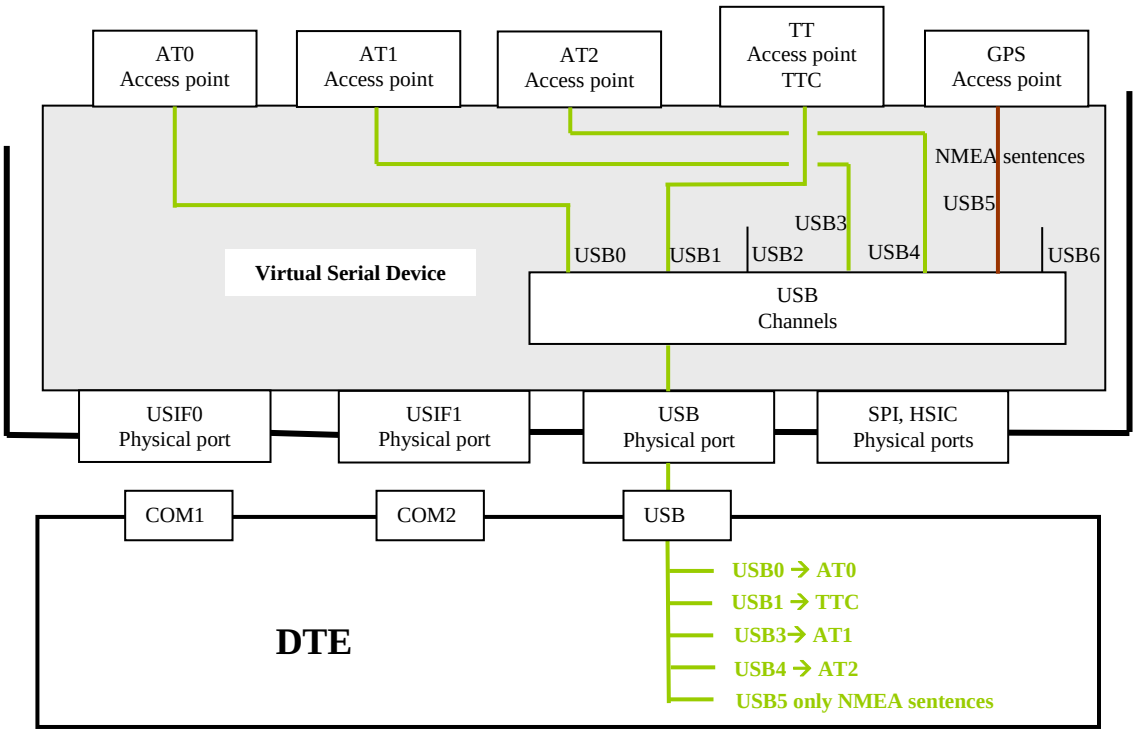

Fig. 28: USB5 Channel Supports Only NMEA Sentences

6.1.2. GPS/NMEA Sentences from External GPS

The external GPS receiver is connected to the module through USIF1 serial port.

6.1.2.1. AT#PORTCFG=11

Tab. 27 summarizes the starting configuration of the module (#PORTCFG=11). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered through USIF0 port. AT0 parser executes the AT commands, and after that NMEA sentences and AT commands run on USIF0 port as summarized in Tab. 39. See also Fig. 29

AT#PORTCFG=11					
	AT0	AT1	AT2	TT	Ext. GPS/NMEA
No USB cable					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				X
USIF1					
SPI					

Tab. 39: USIF1 Port Connected to External GPS

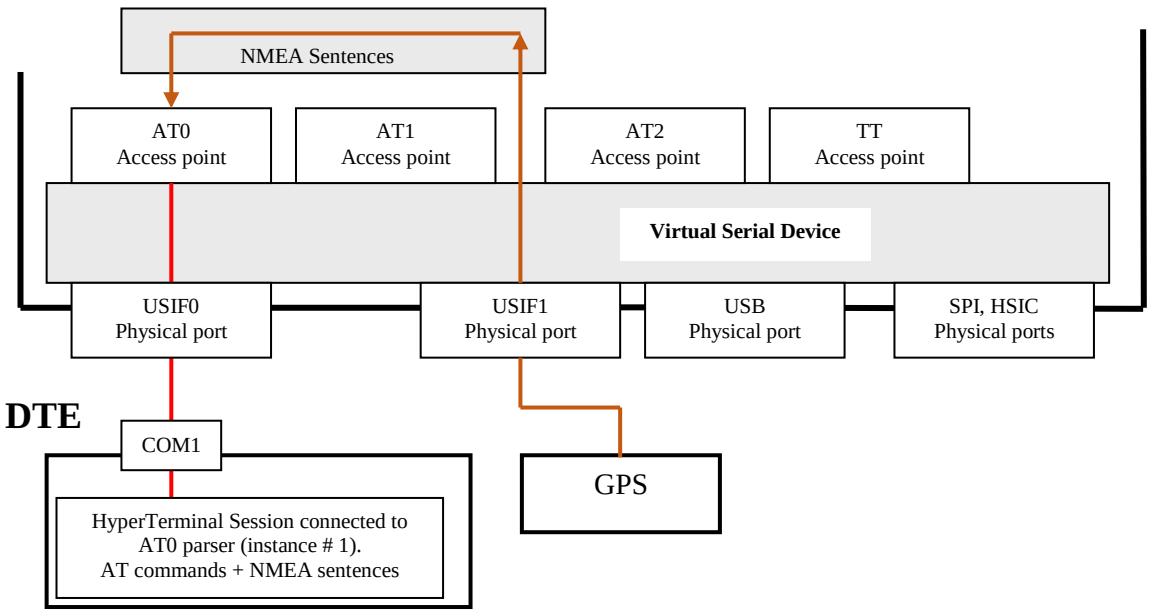

Fig. 29: USIF0 Port Support AT Commands + NMEA Sentences (External GPS)

6.1.2.2. AT#PORTCFG=11 + USB

Tab. 28 summarizes the starting configuration of the module (#PORTCFG=11+USB). Now, enable GPS/NMEA sentences via AT$GPSP=1 and AT$GPSNMUN=1,… AT commands entered, for example, through USB0 channel. AT1 parser executes the AT commands, and after that NMEA sentences and AT commands run on USB0 channel as summarized in Tab. 40. See also Fig. 30.

AT#PORTCFG=11					
	AT0	AT1	AT2	TT	Ext. GPS/NMEA
USB0		X			X
USB1				TTC	
USB2					
USB3			X		
USB4					
USB5					
USB6					
USBHSI0					
USBHSI1					
USBHSI2					
USBHSI3					
USIF0	X				
USIF1					
SPI					

Tab. 40: USIF1 Port Connected to External GPS + USB Cable

NOTICE: the user can issue the AT commands through USIF0/AT0, USB0/AT1, or USB3/AT2 channel/parser. The NMEA sentences are routed respectively on USIF0, USB0 (as shown by the example), or USB3 channel.

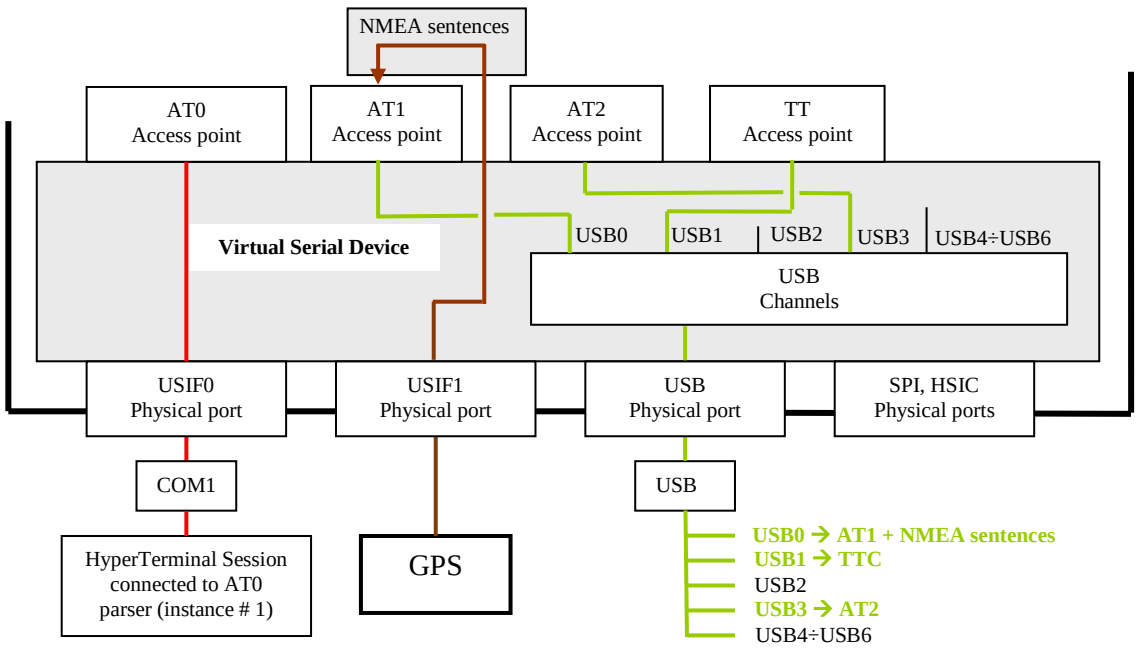

Fig. 30: USB0 Channel Support AT Commands + NMEA Sentences (External GPS)

6.2. Python

In this section, it is assumed that the reader is familiar with Python language. To have information on the software versions supporting different Python modules refer to document [3].

Telit's modules provide the Python programming language to develop control scripts in accordance with user communication and hardware needs. As shown in Fig. 31, the VSD provides two access points called VHWDTE0 and VHWDTE1. MDM and MDM2 Python modules are logically connected respectively to VHWDTE0 and VHWDTE1.

Let us assume that the module is using the ports configuration #PORTCFG=1, summarized on Tab. 7 (factoring setting), no USB cable is plugged in. When the Python script runs the ***import MDM*** instruction, the VSD disconnects the USIF0/AT0 logical connection and sets up the logical connection VHWDTE0/AT0; now, the script can access AT0 parser. In the same way, ***import MDM2*** instruction requires that the VSD sets up the logical connection VHWDTE1/AT1. The Fig. 31 shows that USIF0 is disconnected and cannot be used by an external device.

Python script can use the USIF0 port through the ***import SER*** instruction. The Fig. 32 shows the new connection.

The three Python software modules (MDM, MDM2 and SER) use three independent resources: AT0, and AT1 Access Points, and USIF0 physical port. No resources contention can arise among them. As a rule, we can say that the MDM, MDM2 and SER instructions steal the above-mentioned resources regardless their current owner.

As shown in the next pages there are other Python modules to create logical connection between a physical port and an Access point.

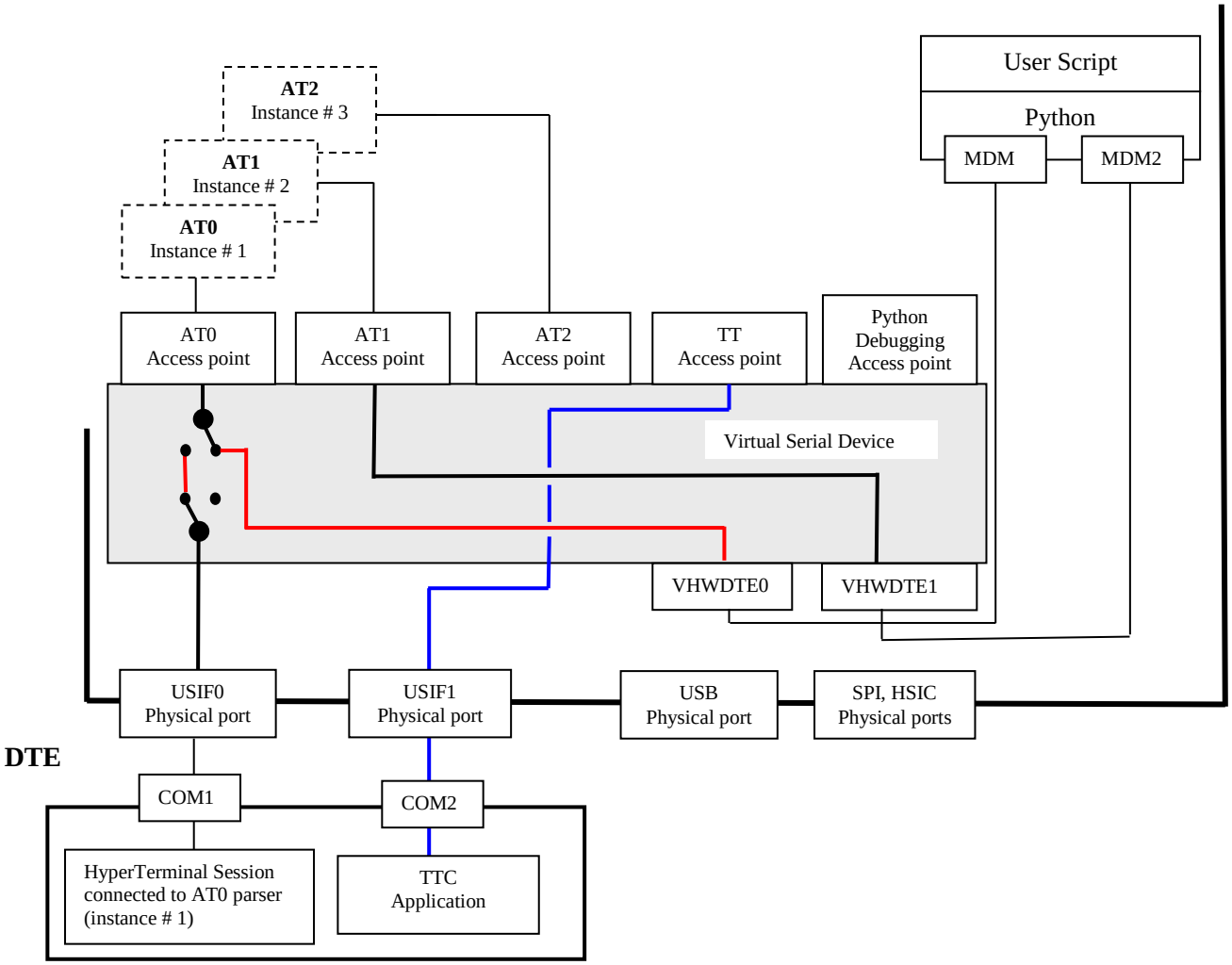

Fig. 31: Python & MDM, MDM2 Modules

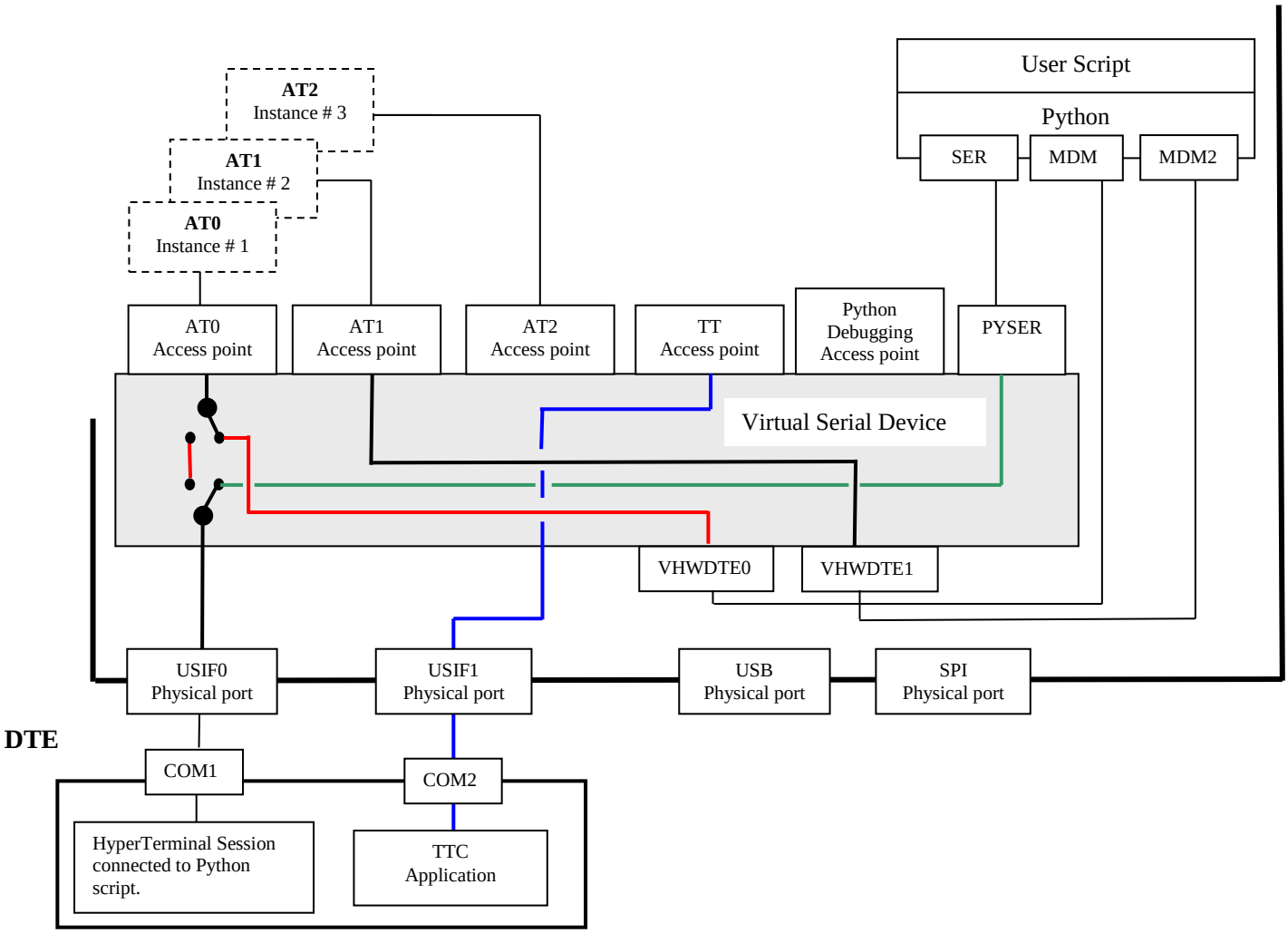

Fig. 32: Python & MDM, MDM2, SER Modules

In accordance with the installed software version, Python script can run the ***import USB0*** instruction to access the USB0 channel by means of PYUSB0 access point. The figure below shows the new connection.

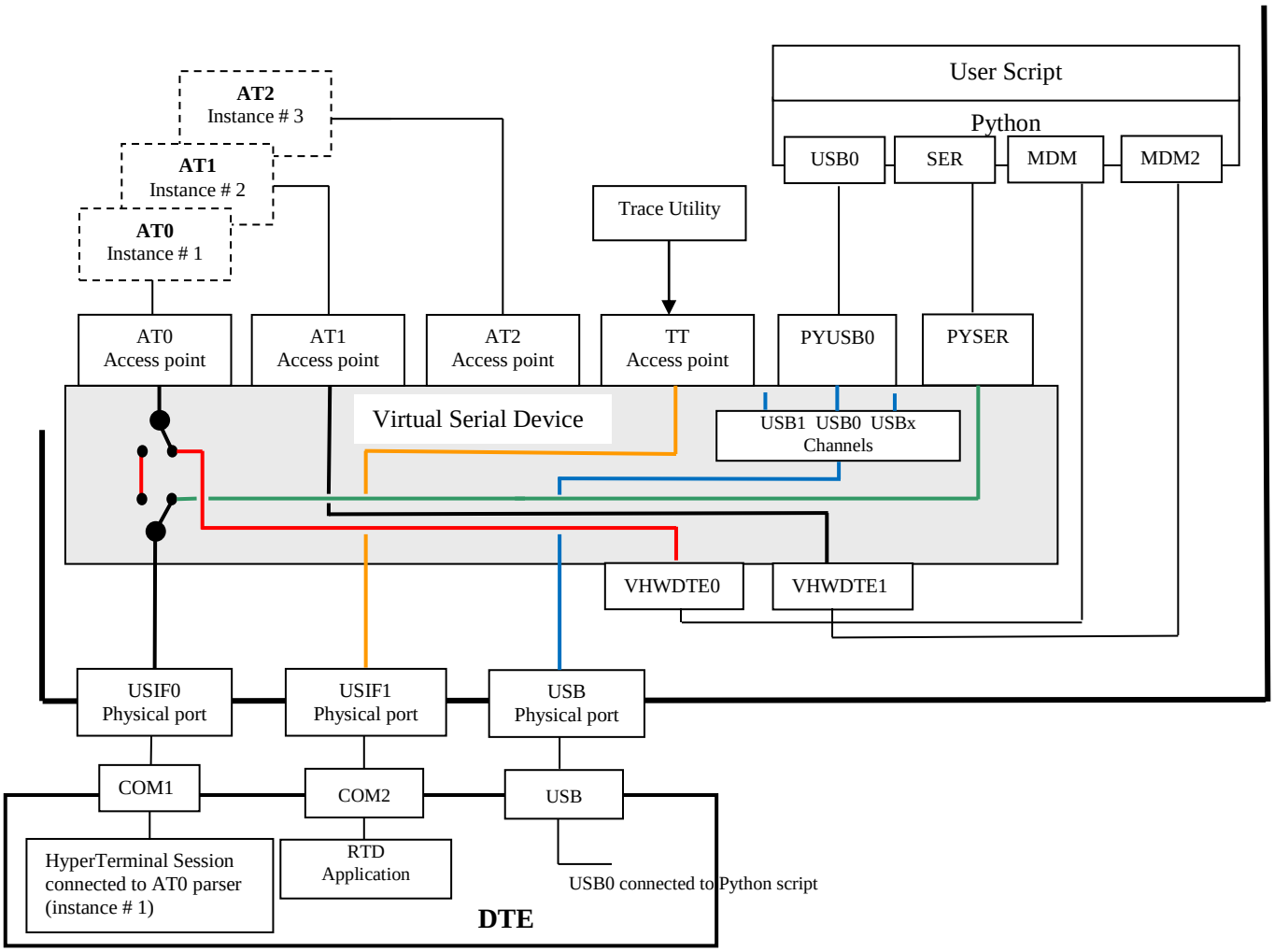

Fig. 33: Python & MDM, MDM2, SER, USB0 Modules

The Python software modules MDM, MDM2, SER, and USB0 use four independent resources: AT0, AT1 Access Points, and USIF0 physical port, USB0 channel. No resources contention can arise among them. As a rule, we can say that the MDM, MDM2, SER, and USB0 instructions steal the resources regardless their current owner.

6.2.1. Python Script Debugging

Let us assume that the user needs to debug a new Python script. To perform the debugging session, the user forces the module into #PORTCFG=3 configuration, refer to Tab. 11. Suppose that the Python script runs: ***import MDM, import MDM2, import SER*** and ***print*** instructions.

The figure below sketches the actions results of the first tree instructions. Moreover, the figure shows that the Python script switches the USIF1 from AT2 Access point to Python Debugging Access point; the print messages are available on USIF1 port.

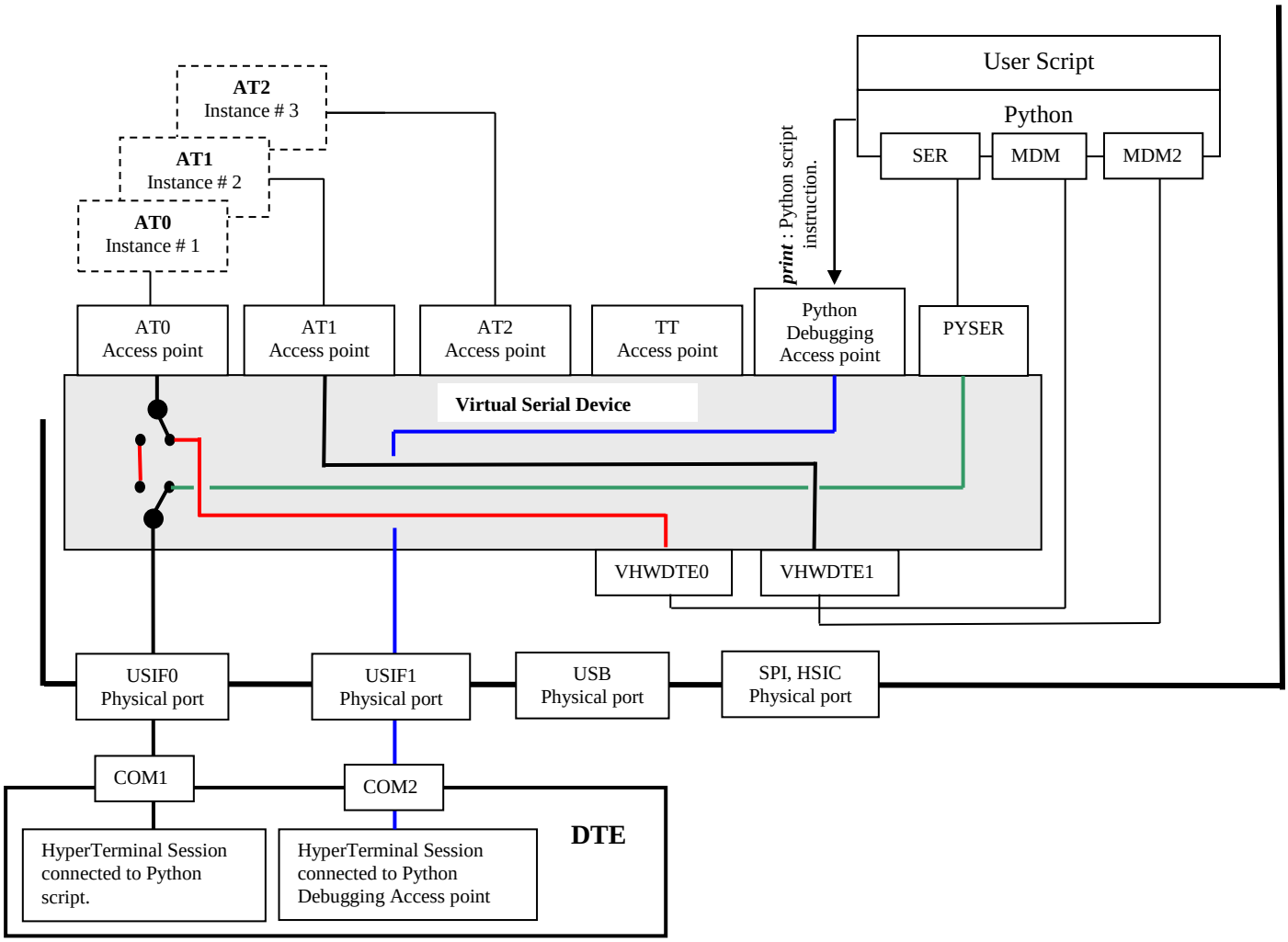

Fig. 34: Python & MDM, MDM2, SER and Print Modules

6.2.2. SER2 Instruction

6.2.2.1. AT#PORTCFG=0

Let us assume that the module is using the ports configuration #PORTCFG=0, no USB cable, refer to Tab. 5. When the Python script runs the instruction ***import MDM***, the VSD disconnects the USIF0/AT0 logical connection and establishes the logical connection VHWDTE0/AT0; now, the script can access AT0 parser. In the same way, ***import MDM2*** instruction requires that VSD sets up the logical connection VHWDTE1/AT1, refer to Fig. 35.

In accordance with the installed software version, refer to document [3], Python script can run the ***import SER2*** instruction to use the USIF1 port through the access point PYSER2. The figure below shows the new connection.

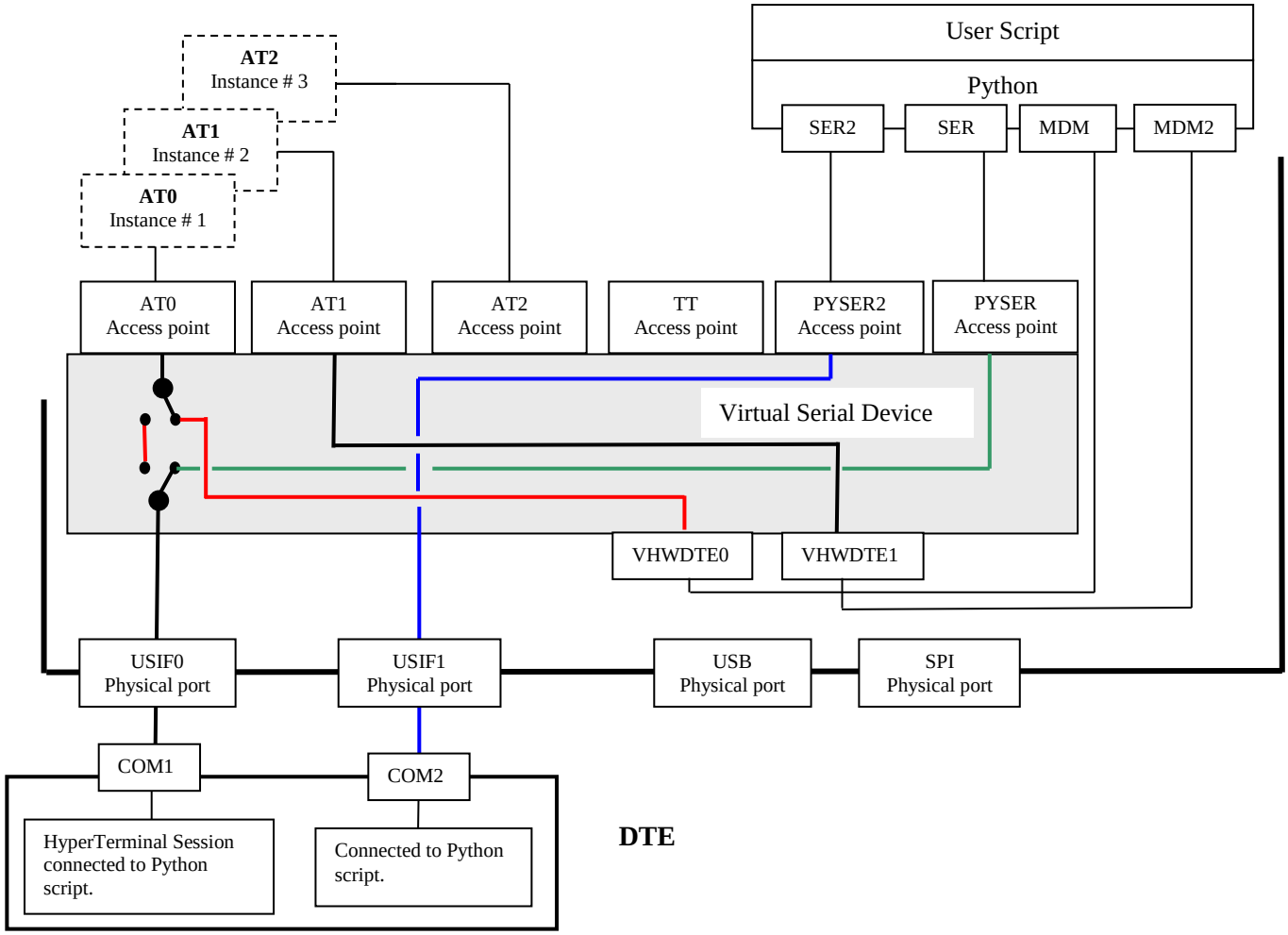

Fig. 35: Python & MDM, MDM2, SER, SER2 Modules

The Python software modules MDM, MDM2, SER, and SER2 use four independent resources: AT0, AT1 Access Points, and USIF0, USIF1 physical ports. No resources contention can arise among them. As a rule, we can say that the MDM, MDM2, SER, and SER2 instructions steal the resources regardless their current owner.

6.2.2.2. AT#PORTCFG=3

Let us assume that the module is using the ports configuration #PORTCFG=3, no USB cable, refer to Tab. 11. When the Python script runs the instruction ***import MDM***, the VSD disconnects the USIF0/AT0 logical connection and establishes the logical connection VHWDTE0/AT0; now, the script can access AT0 parser. In the same way, ***import MDM2*** instruction requires that VSD sets up the logical connection VHWDTE1/AT1, refer to Fig. 36.

In accordance with the installed software version, refer to document [3], Python script can run the ***import SER2*** instruction. In this case, the VSD disconnects the USIF1/AT2 logical connection and sets up the logical connection USIF1/PYSER2; now, the script can access USIF1.

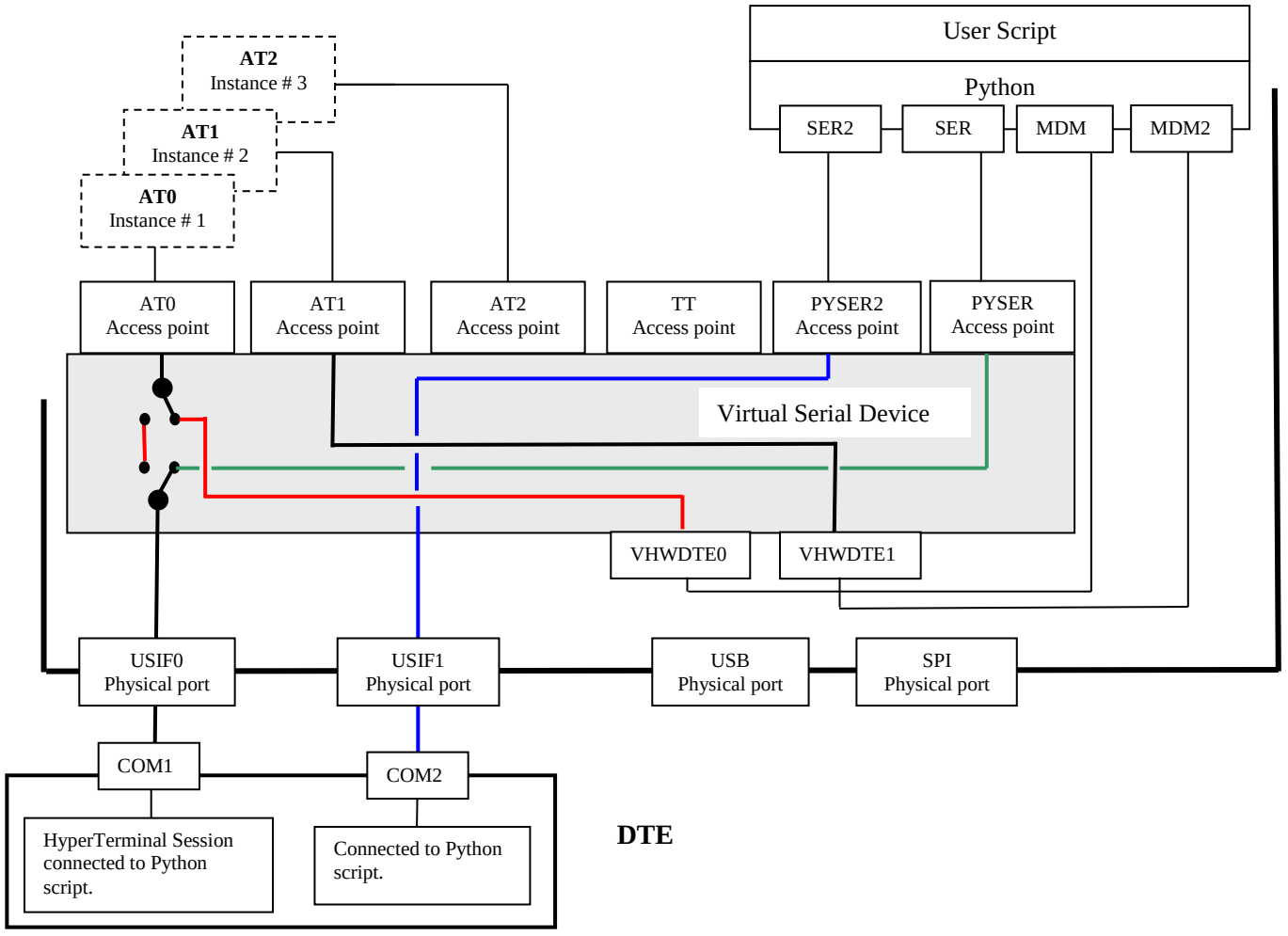

Fig. 36: Python & MDM, MDM2, SER, SER2 Modules

The Python software modules MDM, MDM2, SER, and SER2 use four independent resources: AT0, AT1 Access Points, and USIF0, USIF1 physical ports. No resources contention can arise among them. As a rule, we can say that the MDM, MDM2, SER, and SER2 instructions steal the resources regardless their current owner.

NOTICE: ***print*** instruction, see chapter 6.2.1, and ***SER2*** instruction, both use the USIF1 hardware resource. In case of USF1 contention, ***SER2*** instruction steals USIF1 to ***print***.

6.3. AppZone

To have information on the AppZone layer and its functions (APIs) mentioned in this section, refer to documents [7], and [8].

6.3.1. USIFx Ports

Example 1:

Fig. 2 sketches the starting ports configuration of the module; no USB cable is connected. For example, run a user AppZone Application that does not use neither serial ports nor any ATx parsers; Fig. 37 shows the resulting ports arrangement.

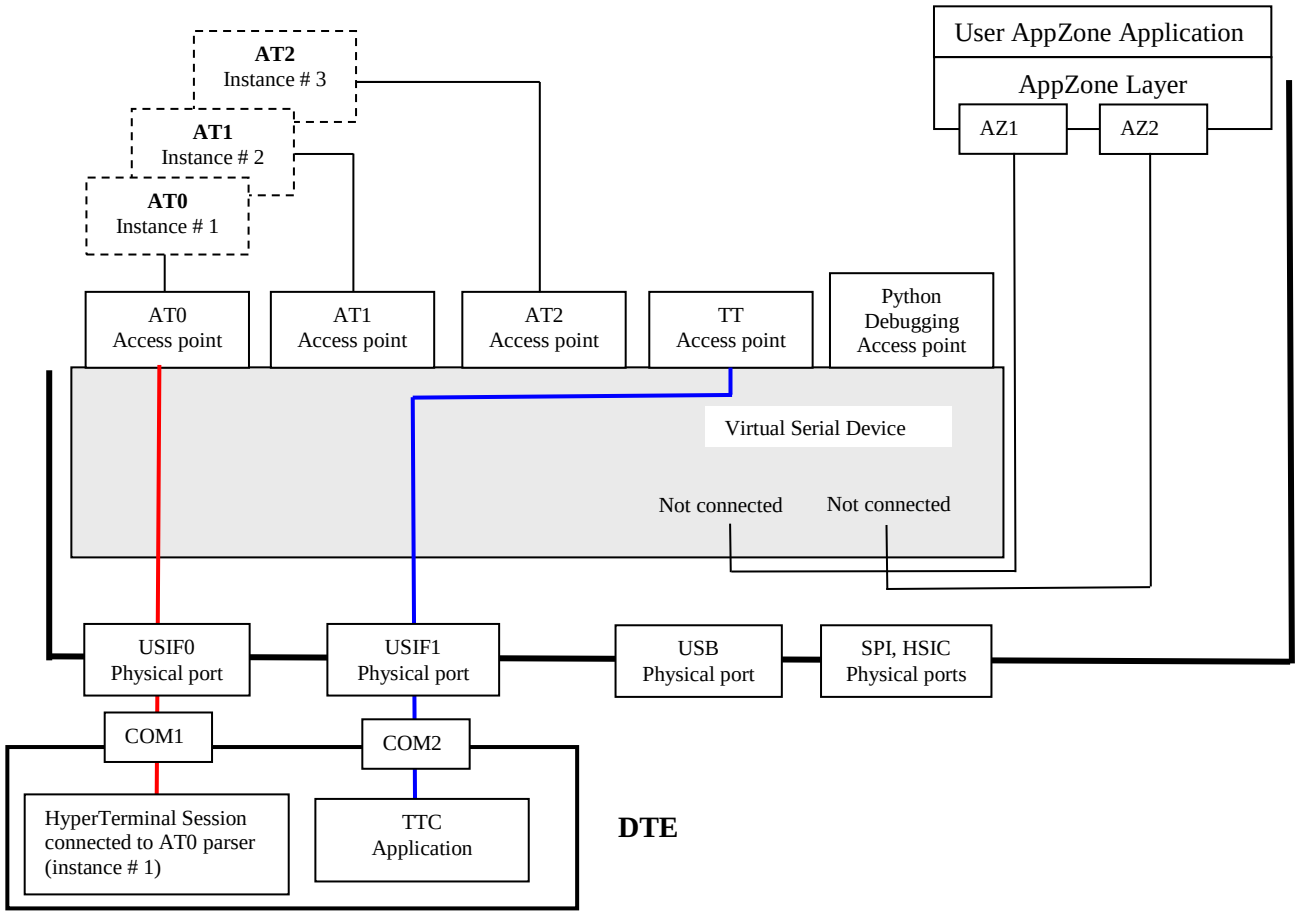

Fig. 37: AppZone Application without Connections

Example 2:

Starting from the configuration of the **Example 1**, use **m2m_os_iat_set_at_command_instance()** function to connect logically the AZ1 and AZ2 access points respectively to AT1 and AT2 parsers, in addition use **PrintToUart()** function to use USIF0 port. Fig. 38 shows the resulting ports configuration.

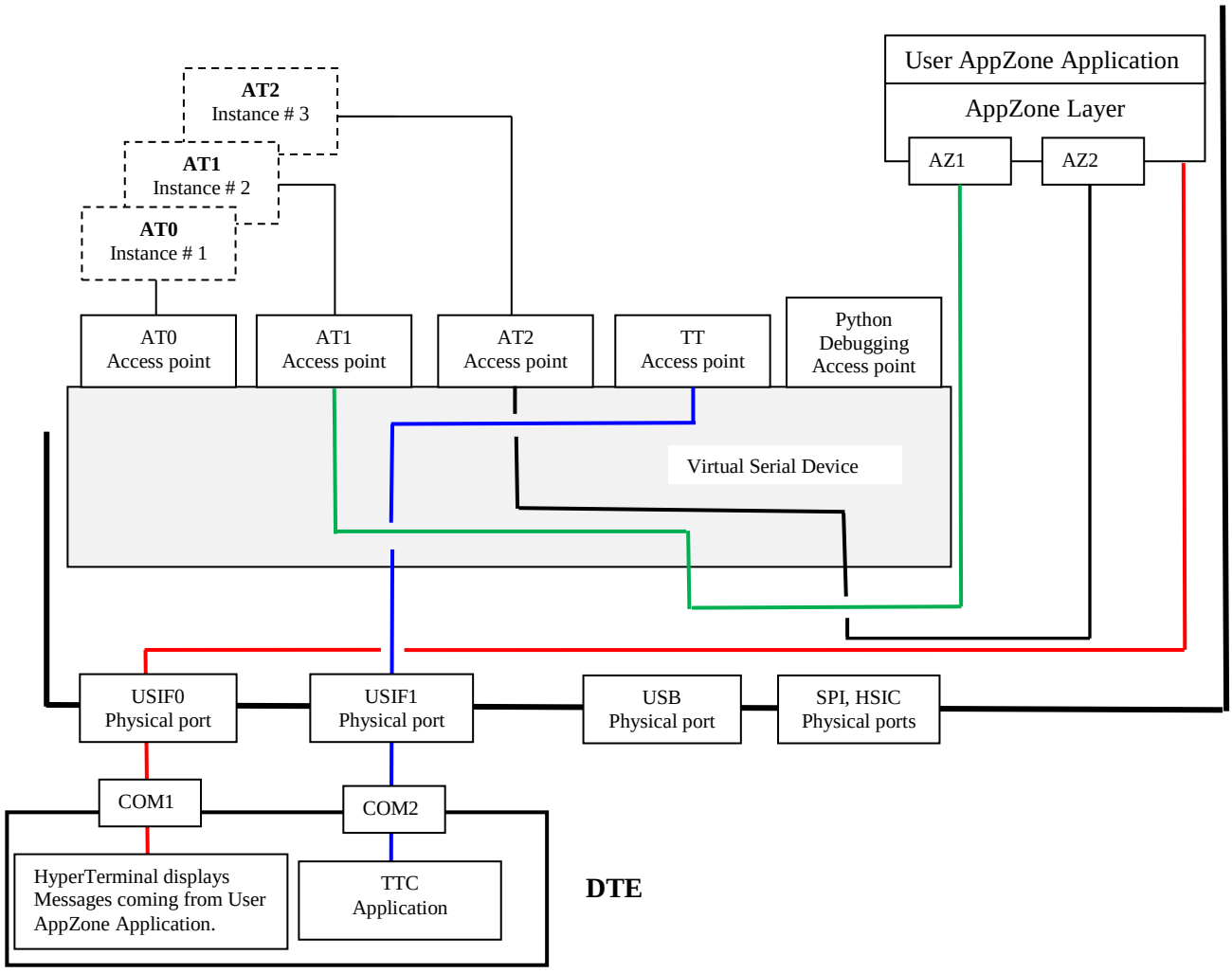

Fig. 38: AppZone Application Connected to AT1, AT2 Parsers, and USIF0 Serial Port

Example 3:

Starting from the configuration of the **Example 1**, use **m2m_hw_uart_ioctl (uart_fd, M2M_HW_UART_IO_AT_MODE_SET, M2M_HW_UART_IO_AT_MODE_ON)** function to route data, received from USIF0, to AT1 parser. Fig. 39shows the resulting ports configuration.

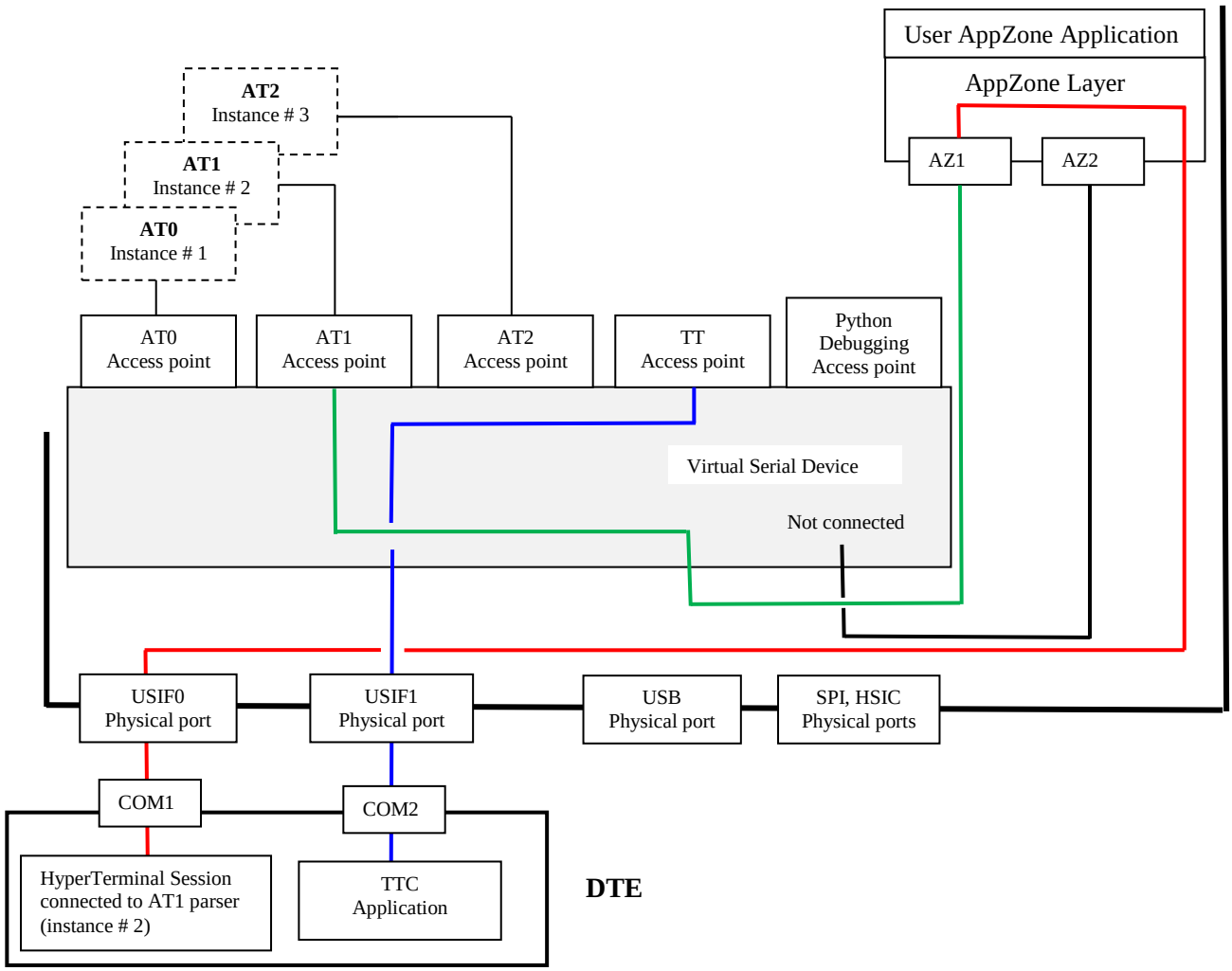

Fig. 39: USIF0 Connected to AT1 Parser through AppZone Layer

6.3.2. USB Channels and Instances

Example 1:

Fig. 2 sketches the starting configuration of the module; no USB cable is connected. Now, connect the USB cable to the module. The module recognizes the “plug in” event and assumes the factory ports arrangement depicted in Fig. 5. Tab. 8 summarizes the new factory configuration. USB0÷USB6 are the seven channels provided by the USB port.

NOTICE: starting the user AppZone Application from the factory setting, only two USB channels are available for the user application.

Use:
m2m_hw_usb_open(USB_CH0, handle0) to disconnect USB0 channel from AT1 parser, connect it to AppZone layer, and get its handle. See Fig. 40

m2m_hw_usb_open(USB_CH3, handle3) to disconnect USB3 channel from AT2 parser, connect it to AppZone layer, and get its handle. See Fig. 40

NOTICE: m2m_hw_usb_open() returns the control to the calling task only if the USB cable is connected.

Manage USB channels:
m2m_hw_usb_read or **m2m_hw_usb_write** to read or write data from/to USB0 and USB3 channels.

The table below summarizes the relationship among the following items:

USB instance: used to manage, for example, lock and unlock among USB channels, see **USB_print()** function included in **M2M_ulilities.c** file.
handle: used to manage the USB channel
channel: used to identified the name of the USB channel

USB instance refer to: m2m_hw_usb_get_instance	handle refer to: m2m_hw_usb_open	USB channel name refer to: m2m_hw_usb_getch_from_handle
USER_USB_INSTANCE_T enum	handle	M2M_USB_CH enum
USER_USB_INSTANCE_T enum	handle	M2M_USB_CH enum
USER_USB_INSTANCE_T enum	handle	M2M_USB_CH enum

Tab. 41: USB Instances, Handles, and Channels

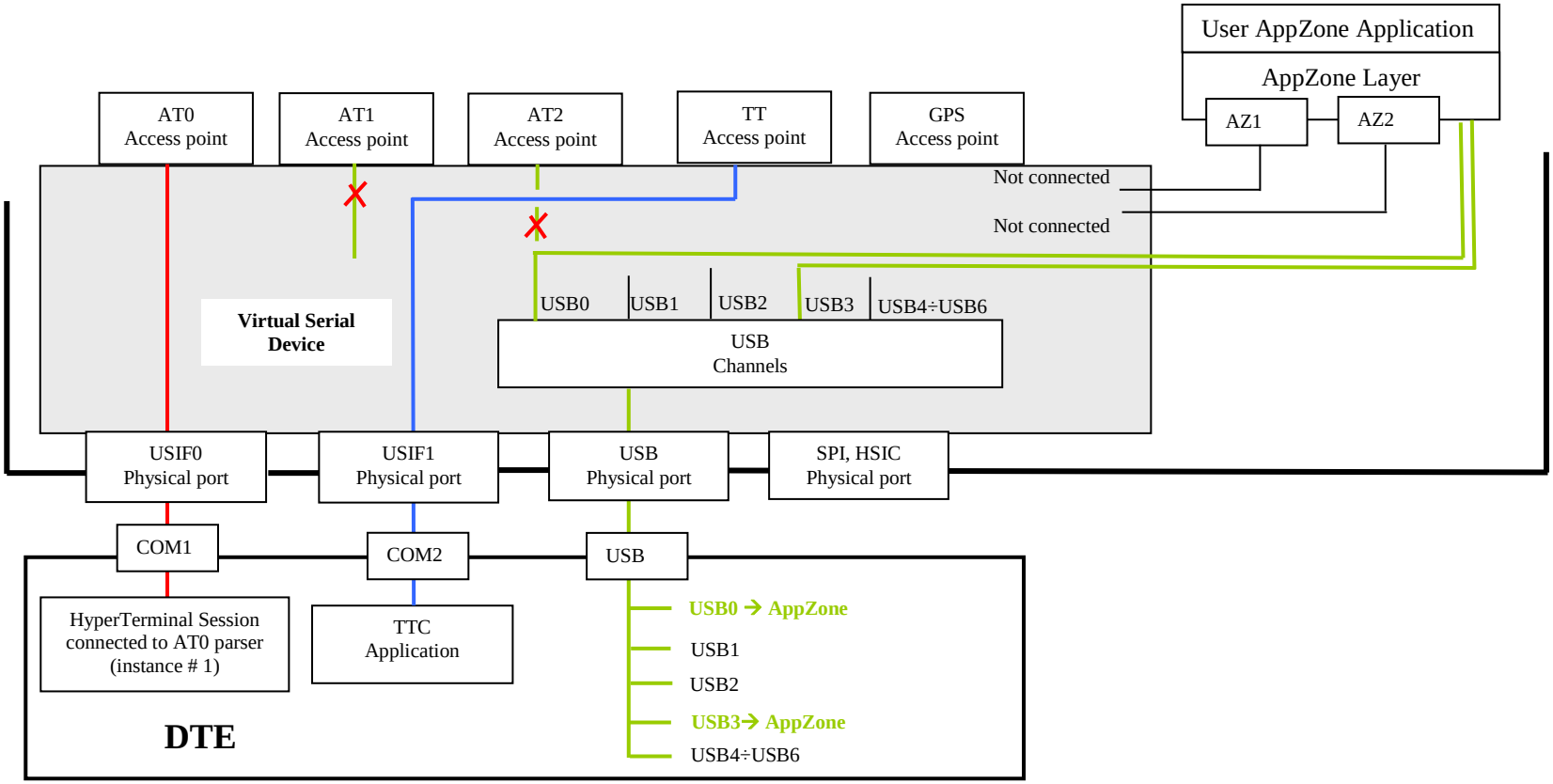

Fig. 40: USB0 and USB3 Channels Available to AppZone Application

Example 2:

Tab. 22 summarizes the starting ports configuration of the module.

NOTICE: starting the user AppZone Application from the configuration shown in the Tab. 22, three USB channels are available for the user application: USB0, USB3, and USB4. USB1 is reserved for TTC.

Use:
m2m_hw_usb_open(USB_CH0, handle0) to disconnect USB0 channel from AT0 parser, connect it to AppZone layer, and get its handle. See Fig. 41.

m2m_hw_usb_open(USB_CH3, handle3) to disconnect USB3 channel from AT1 parser, connect it to AppZone layer, and get its handle. See Fig. 41.

m2m_hw_usb_open(USB_CH4, handle4) to disconnect USB4 channel from AT2 parser, connect it to AppZone layer, and get its handle. See Fig. 41.

Manage USB channels:
m2m_hw_usb_read or **m2m_hw_usb_write** to read or write data from/to USB0, USB3, and USB4 channels.

Example 3:

Starting from the configuration summarized in Tab. 22 and shown in Fig. 12, use **m2m_hw_usb_ioctl (handle4, M2M_USB_AT_MODE_SET, M2M_HW_USB_IO_AT_MODE_ON)** function to route data received from USB4 channel to AT1 parser. Fig. 42 shows the resulting ports configuration.

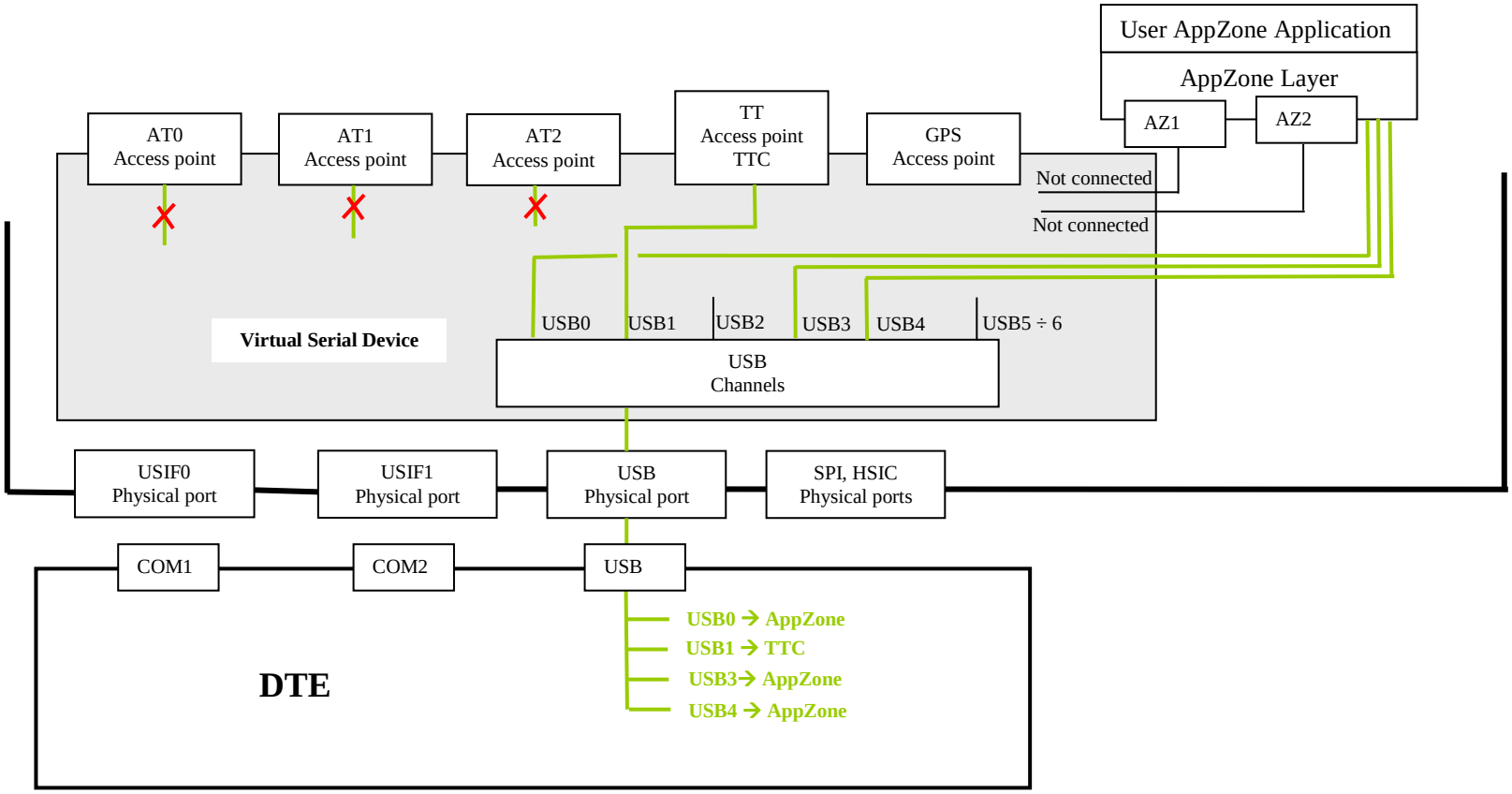

Fig. 41: USB0, USB3, and USB4 Channels Available to AppZone Application

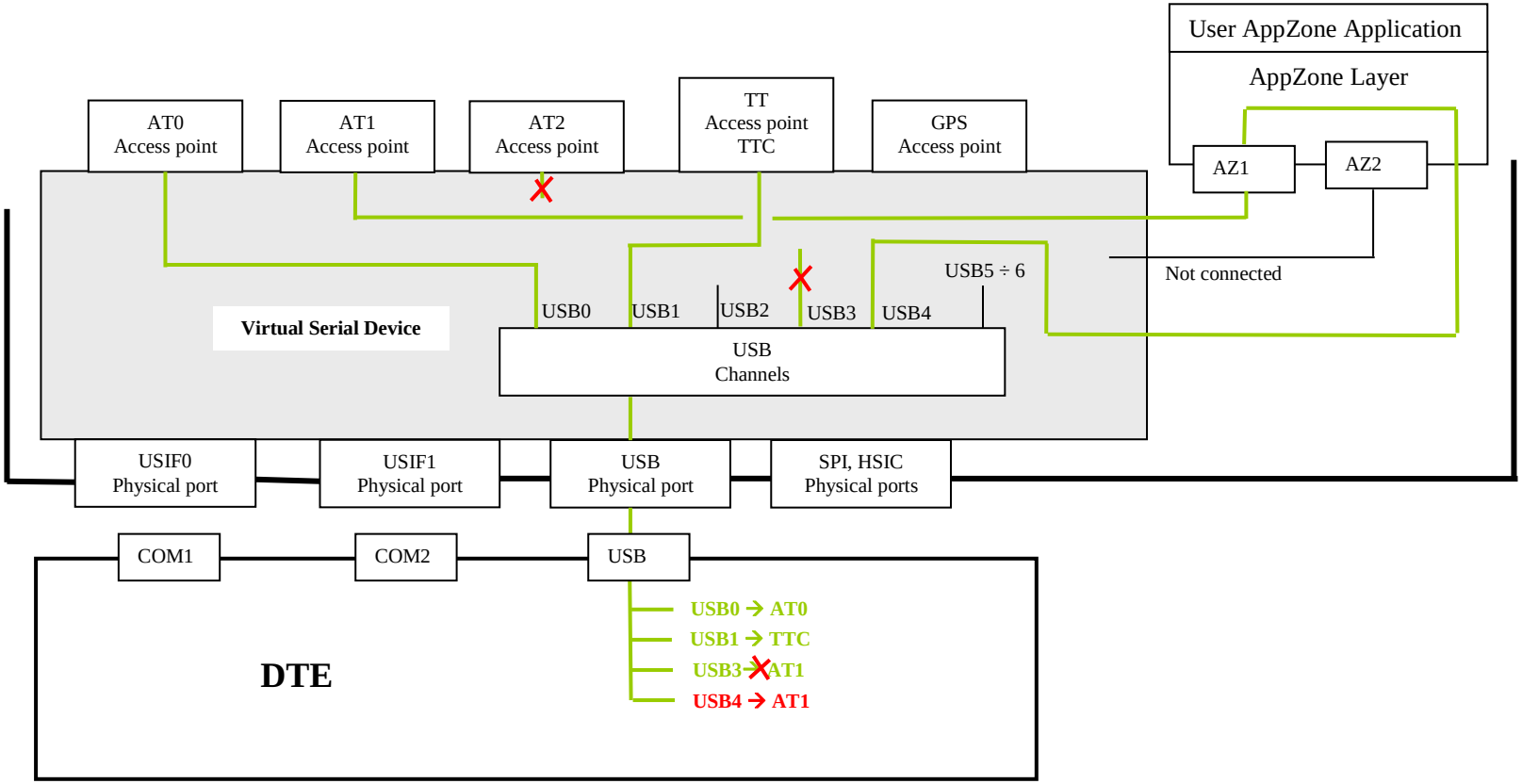

Fig. 42: USB4 Channel connected to AT1

7. The Winning Ports Configuration

Here are two examples showing that the last port configuration set by the user overrides the previous one.

There are two ways to change module ports arrangement in addition to use AT#PORTCFG command:

- Plug in/out the USB cable;
- Enter the AT+CMUX=0 command.

NOTICE: TELIT Serial Port MUX application sends automatically the AT+CMUX=0 command to the module, see chapters 5.1, and 5.2

Example 1

Module: Tab. 5 summarizes module ports configuration.

User action: user runs the TELIT Serial Port MUX application on Windows-PC; the application connects logically Virtual Ports COM20÷COM23 to COM1.

PC: it provides the required Virtual Ports. When the user starts an application (e.g. Hyper Terminal) connected to one of the three Virtual Ports (the fourth one is spare), TELIT Serial Port MUX application sends the AT+CMUX=0 command to the module.

Module: in accordance with the received command, the involved AT Parser starts the CMUX protocol. The module enters the configuration shown on Fig. 19.

User action: now, the user connects USB cable.

Module: it enters the configuration shown on Fig. 4.

PC: it provides seven new virtual “COM” connected logically to the seven USB channels. The CMUX protocol is disabled, and the TELIT Serial Port MUX application running on Windows PC is no more connected to the module, it should be closed. COM1 is ready for new applications (e.g. Hyper Terminal).

User action: now, the user disconnects USB cable.

Module: it enters again the configuration shown in Tab. 5

Example 2

Module: Tab. 5 summarizes module ports configuration.

User action: user connects USB cable.

Module: in accordance with the user action, the module enters the configuration shown in Fig. 4.

PC: it provides seven virtual “COM” required by USB drivers to connect logically the seven USBx channels.

User action: user runs the TELIT Serial Port MUX application on the Windows PC; the application connects logically Virtual Ports VCOM20 ÷ VCOM23 to USB3→VCOM8 channel.

PC: it provides the required Virtual Ports. When the user starts an application (e.g. Hyper Terminal) on a Virtual Ports, TELIT Serial Port MUX sends the AT+CMUX=0 command to the module.

Module: in accordance with the received command, the involved AT Parser starts the CMUX protocol. The module enters the configuration shown in Fig. 22.

User action: now, the user disconnects USB cable.

Module: it enters again the configuration shown on Tab. 5.

PC: discards the seven virtual “COM” connected logically to the seven USBX channels. The CMUX protocol is disabled, TELIT Serial Port MUX application running on Windows PC is no more connected to the module, and it should be closed.

The two examples show that the last required port configuration overrides the previous one.

8. Abbreviation and Acronyms

3G Tool	Third Generation Trace Tool (for internal use only)
DTE	Data Terminal Equipment
GPS	Global Positioning System
HSIC	USB High Speed Inter-Chip Interface
NMEA	National Marine Electronics Association
PPP	Point to Point Protocol
SPI	Serial Peripheral Interface
TTC Tool	Telit Trace Client Tool
USIFx	Universal Serial Interface
VSD	Virtual Service Device

9. Document History

Revision	Date	Product/SW Version	Changes
0	2011-11-23	/	First issue
1	2011-11-28	/	Mobile Analyzer changed in Trace Tool (Generic TT)
2	2012-01-27	/	Updated the Applicability Table AT+CMUX=1 changed in AT+CMUX=0
3	2012-02-28	/	Updated parameter range of AT#PORTCFG Command Modified the SPI Physical port connection on all figures Introduced TTC and 3G (see TT)
4	2012-07-03	/	Added PORTCFG=7 and updated PORTCFG tables. Modified document title “HE Family Ports Arrangements” in “HE910 Family Ports Arrangements”. General review of the entire document.
5	2012-09-17	/	Added PORTCFG=8. Updated Applicability Table.
6	2013-07-29	/	The document title has been changed from “HE910 Family Ports Arrangements” to “HE910/UE910 Families Ports Arrangements” Updated Applicability Table, and some figures. Rearranged GPS chapters and modified some names of chapters. Added the note about +CFUN command in chapter 2. New features supported from SW version 12.00.004: PORTCFG=9, PORTCFG=10, and HSIC physical port. The factory setting has been changed from #PORTCFG=0 to #PORTCFG=1.
		Added products: UE910-EUR/EUD/12.00.xx4 UE910-NAR/NAD/12.00.xx4	/
7	2014-02-24	/	Added some figures and related descriptions.
8	2014-02-28	/	The document title has been changed from “HE910/UE910 Family Ports Arrangements” to “HE910/UE910/UL865 Families Ports Arrangements”
		Added products: UL865-EUR/EUD/12.00.xx4 UL865-NAR/NAD/12.00.xx4 UL865-N3G/12.00.xx4	/
9	2015-02-16	Updated Applicability Table **HE910 Family** HE910 12.00.xx6 HE910-D 12.00.xx6 HE910-EUR / HE910 EUD 12.00.xx6 HE910-EUG / HE910-NAG 12.00.xx6 HE910-NAR / HE910-NAD 12.00.xx6 **UE/UL Family (Embedded)** UE910-EUR / UE910-EUD 12.00.xx6 UE910-NAR / UE910-NAD 12.00.xx6 UL865-EUR / UL865-EUD 12.00.xx6 UL865-NAR / UL865-NAD 12.00.xx6 UL865-N3G 12.00.xx6	Chapters’ organization has been rearranged; some chapters have been reviewed or removed. Added: Services Coexistence Table, AppZone Service, AT#PORTCFG=11, and AT#PORTCFG=12.