

Introduction

This document is a description of the firmware and GUI interface developed to support the application involving L9301 device. The first part of the document is focused to depict the data structure used for the SPI communication between L9301 device (slave) and the microcontroller (master). The second part of the document is dedicated to describe the Graphical User Interface and the L9301 functionalities embedded in the GUI software. The last part is devoted to a step-by-step description of the installation of firmware and GUI interface.

Contents

1	L9301 firmware and GUI interface	3
2	Data structure	5
3	Firmware and serial communication	8
4	USART serial protocol	9
5	Firmware main	10
6	State machine	11
7	PWM driving	13
8	SPI driving	17
9	GUI general user interface labview	23
10	SPC560P–DISP: USB drivers installation	32
11	Bibliography	37
12	Revision history	38

1 L9301 firmware and GUI interface

In order to interface L9301 demo board with the loads of the application (such as coils and valves of the ABS/ESP system hydraulic modulator), a data structure has been conceived.

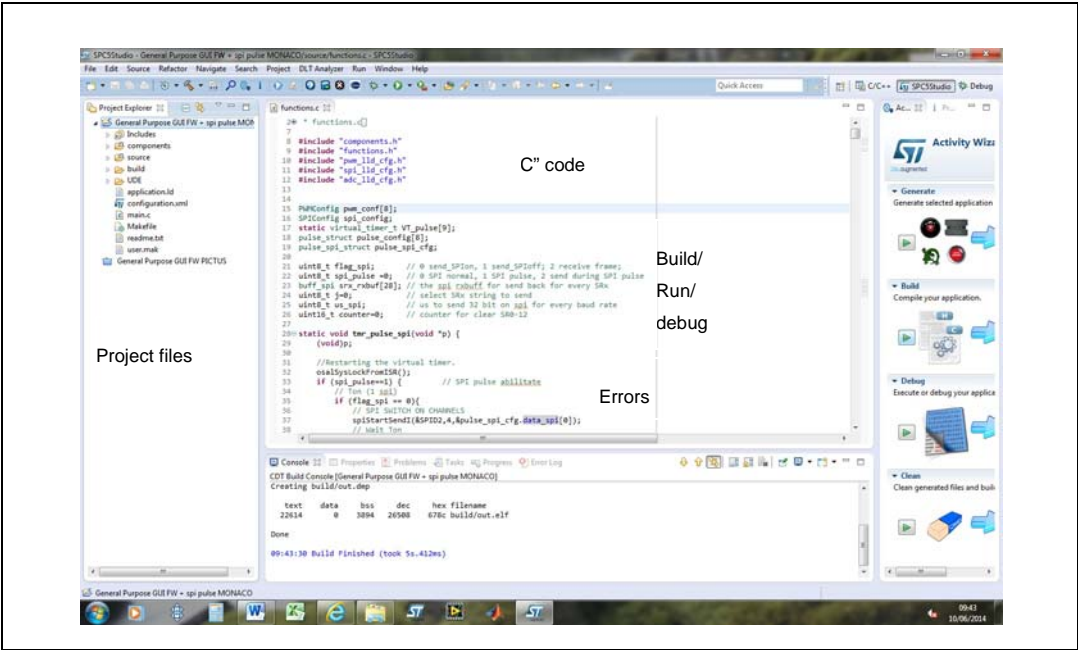
From PC side, a labview interface has been designed and developed. This labview interface allows the PC to communicate with L9301 device by means of MCU SPC560P50 (see [Figure 1](#)).

Figure 1. Demo board SPC560P50



On MCU platform, a general purpose firmware has been developed. This software layer has been done in SPC5 studio environment (see [Figure 2](#)) and has been loaded on the MCU platform by SPC5-UDESTIK linked with mini-USB wire to the PC.

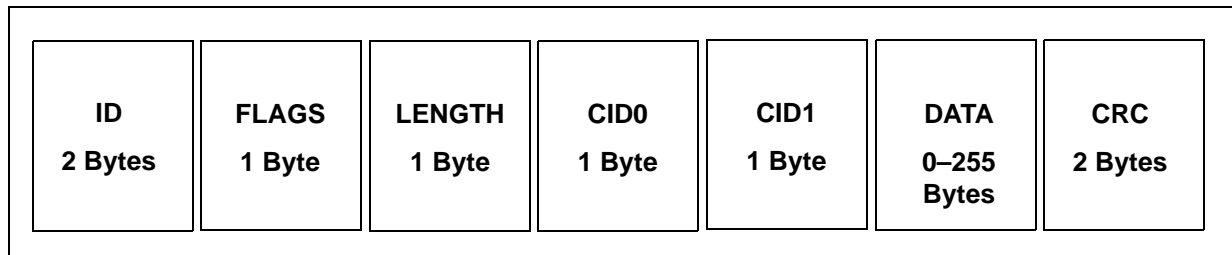
Figure 2. SPC5 Studio



2 Data structure

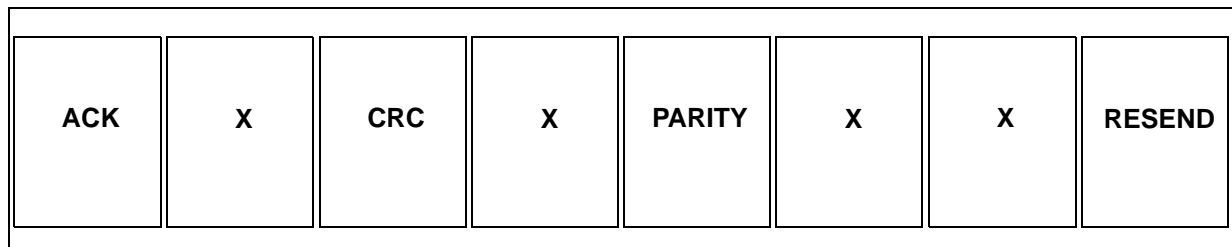
In order to allow the communication between GUI and MCU with a low probability of faults, a variable length data structure (frame) has been defined as shown in [Figure 3](#):

Figure 3. Data structure field bit (frame) MCU



1. ID: used to identify device under test (for L9301 this code corresponds to 0x93 0x01)
2. FLAGS: used to communicate different data as shown in [Figure 4](#):

Figure 4. 8 bits of FLAGS



3. ACK: dedicated to enable/disable the acknowledge send (feedback of acquired data)
4. CRC: dedicated to enable/disable CRC send
5. PARITY: devoted to enable/disable parity check
6. X: case insensitive
7. LENGTH: indicates the DATA field length (in byte)
8. CID0 – CID1: used to identify the type of operation behind the frame to be sent and the contents of the DATA field. In other words, MCU checks periodically the input frame and using a switch-case structure carries out the following configurations and activities as described in [Table 1](#)

Table 1. Firmware case structure

CID0	CID1	Mode/Data	Values	Description
0x01				SPI
	0x00	conf periph		SPI INIT
		data[0]	0x00	ctar cpol
			0x01	
		data[1]	0x00	ctar cpha
			0x01	
		data[2]	0x00	10 Mhz
			0x01	8 Mhz
			0x02	5 Mhz

Table 1. Firmware case structure (continued)

CID0	CID1	Mode/Data	Values	Description
			0x03	4 Mhz
			0x04	2 Mhz
			0x05	1 Mhz
			0x06	0.5 Mhz
			default	1 Mhz
	0x01	normal		SPI SEND –RECEIVE
		data[]	spitxbuf[]	
				rxbuf[]
	0x0C	normal		SPI PULSE
		data[0], data[1]	us	Ton
		data[2], data[3]	us	Toff
		data[4], data[5], data[6], data[7]		channels on
		data[8], data[9], data[10], data[11]		channels off
0x02				PWM
	0x00	conf periph		PWM INIT
		data[0]	0x01	PWM ACTIVE H
			0x02	PWM ACTIVE L
		data[1]	0x00	CH0
		
			0x07	CH7
	0x0C	normal		PWM FREQUENCY
		data[0], data[1]	1	1
			3E8	1000
			7D0	2000
			2710	10000
		data[2]	0x00	CH0
		
			0x07	CH7
	0x0D	normal		PWM DUTY
		data[0], data[1]	0	0
			64	1
			3E8	10
			1388	50

Table 1. Firmware case structure (continued)

CID0	CID1	Mode/Data	Values	Description
			1D4C	75
			2710	100
		data[2]	0x00	CH0
		
			0x07	CH7
0x10				State transition
	0x00	conf com		
	0x01	conf periph		
	0x02	diag		
	0x03	normal		
	0x04	stop		

DATA: can include information described in [Table 1](#) or the 32 bits of device configuration in the case CID0=0x01 and CID1=0x01. It is used to send data toward the device and at the same time to receive data for the analysis.

CRC: acronym of Cyclic Redundancy Check. It is used to check the compliance of the frame sent or received. It is in a one-to-one correspondence with the single frame. It is calculated by a XOR chained on two pieces of 2 bytes of the sent or received frame. The control is done both on GUI and MCU side.

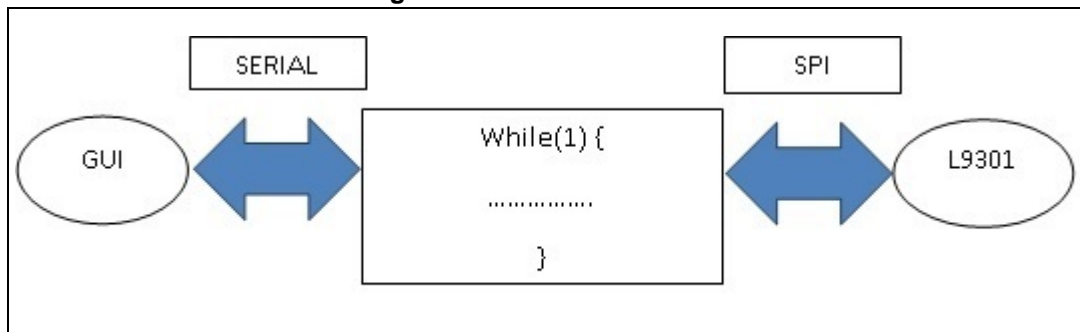
3 Firmware and serial communication

The aims of the MCU are:

- Receive and send data from and to the GUI using a serial protocol
- Send and receive data from and to the device (L9301) by SPI communication protocol
- Generate PWM signals to drive ABS channels

Refer to [Figure 5](#).

Figure 5. GUI – MCU – L9301



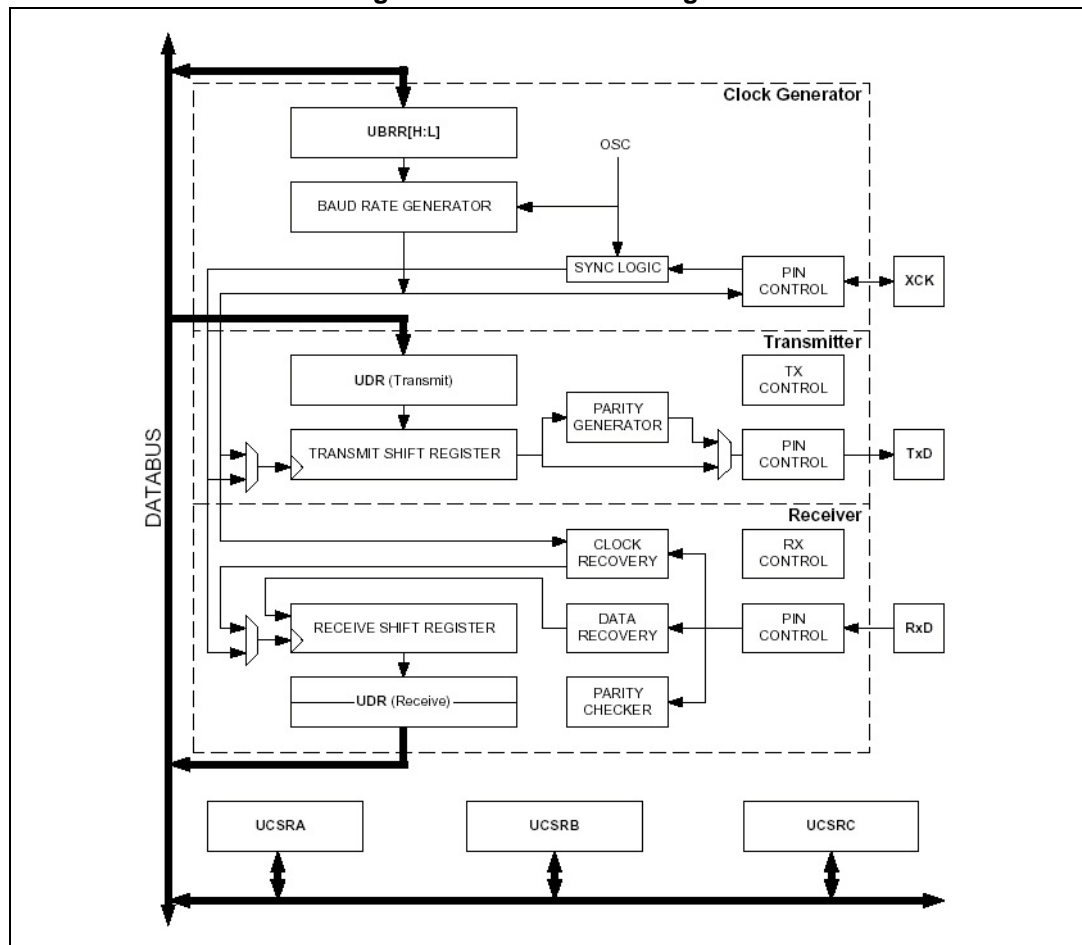
In the first part of the firmware the MCU peripherals USART, SPI and PWM are initialized.

4 USART serial protocol

USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) is a serial communication protocol that allows to interface labview environment with the MCU platform through COM serial port (see [Figure 6](#)). It is composed of three signals:

- Clock Generator
- Transmitter
- Receiver

Figure 6. USART block diagram



Clock Generator is characterized by a logic that allows to use synchronous mode and by a baud rate generator. XCKn (Transfer clock) pin is just used for the synchronous mode.

Transmitter is composed of a write buffer, a serial shift register, a parity generator and of a control logic to manage different serial frames. The write buffer allows a continuous transfer without delays.

Receiver is the most complex part of the USART module. This complexity is due to its clock and units devoted to data recover. These last components are used to receive data in asynchronous mode.

USART has been configured with a baud rate = 115200.

5 Firmware main

The firmware main includes some initializations and a while cycle as shown in [Figure 7](#):

Figure 7. main: while cycle

```
while (1) {  
    check_serial(&frame_received, &valid_frame);  
    if(valid_frame==1){  
        check_data_send_ack (&frame_received, &ack_sent);  
        process_command(&frame_received, &micro_state);  
        valid_frame=0;  
    }  
}
```

When a software structure, as shown in [Figure 7](#) receives a frame, the actions listed below follow:

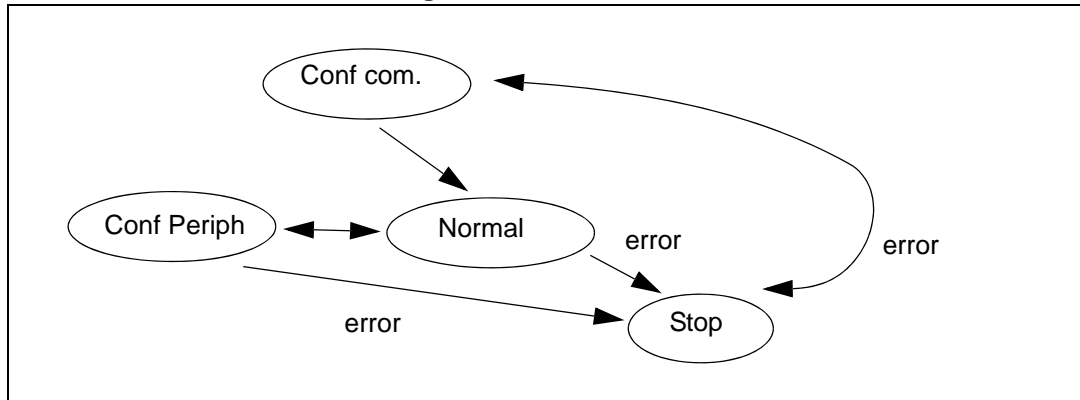
- "check_serial()": checks the compliance of the device (L9301) ID and set the flag "valid_frame" equal to 1 (correct frame)
- "check_data_send_ack()": controls the frame and if it is correct then it transmits to the PC an "ACK OK" (a feedback byte)
- "process_command()": is the core function of the overall firmware and starts when the previous checks are completed

By this function it is possible addressing CID0 and CID1 to enable some set/control on PWM, SPI, GPIO, ADC peripherals.

6 State machine

MCU has been depicted as a state machine. A brief description is reported in [Figure 8](#).

Figure 8. State machine



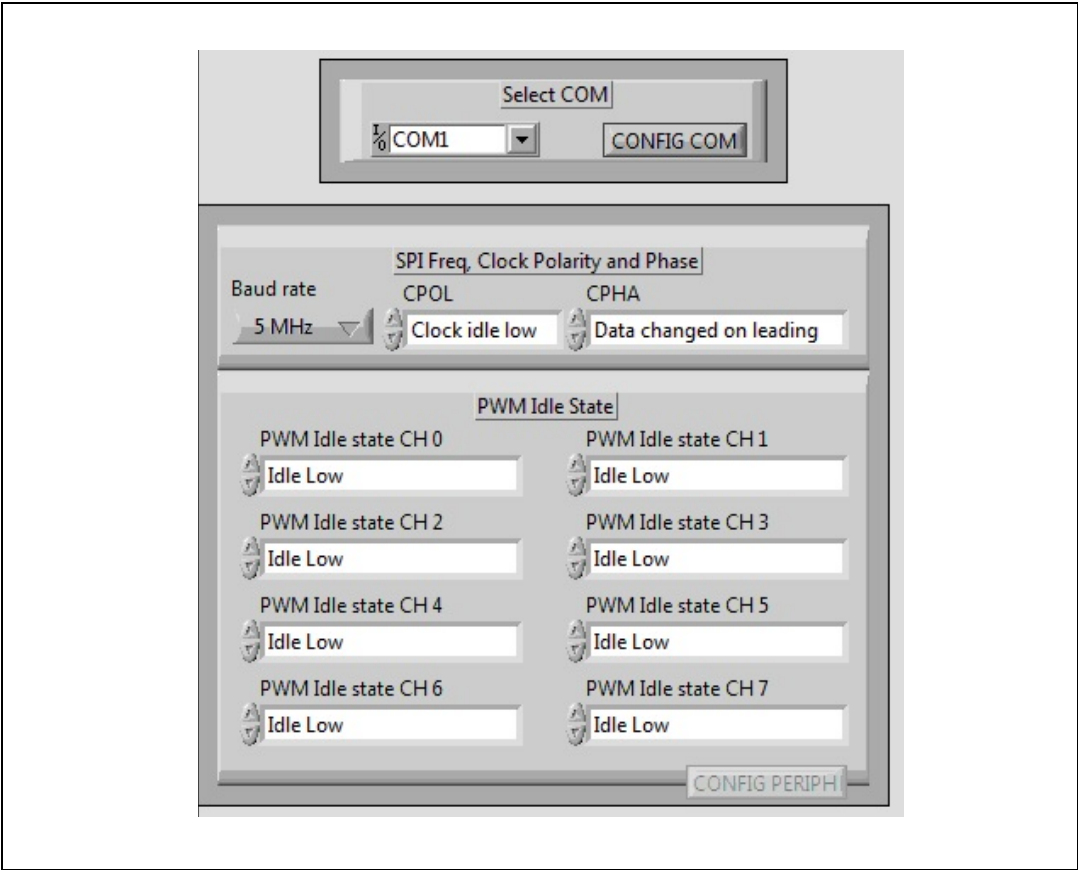
- “Conf com.”: is the initial state. It configures the USART PC/MCU communication
- “Conf Periph”: used to configure the MCU peripherals (SPI, PWM, GPIO, ADC)
- “Normal”: if the configuration is correct, the MCU goes into this state where it can do all the functions, such as SPI send/receive, PWM generation, GPIO, ADC read.
- “Stop”: resets MCU. In this case, MCU can go in an error case or stop itself
- “Conf Periph”: in this state, SPI and PWM initialization occur

SPI module can be set as described below:

- “baud rate” (data transfer speed) of SPI is set at default value of 1 MHz, with the possibility to change on a grid of predefined values in the range 500 kHz and 10 MHz. For the L9301 the maximum value that can be set is 5 MHz
- CPOL: clock polarity, that is, high–active or low–active (for L9301 high–active)
- CPHA: clock phase, that is, when the data has to be read (for L9301 data are read on the falling edge of the SPI–clock)

MCU can set simultaneously up to 8 PWM channels. For each PWM channel it is possible to set the idle state, that is if PWM is low–active or high–active (0 V or 5 V). Both SPI INIT and PWM INIT can be set on the GUI side in Labview in the window “Config Periph” (see [Figure 9](#)).

Figure 9. SPI_INIT and PWM_INIT on the GUI interface

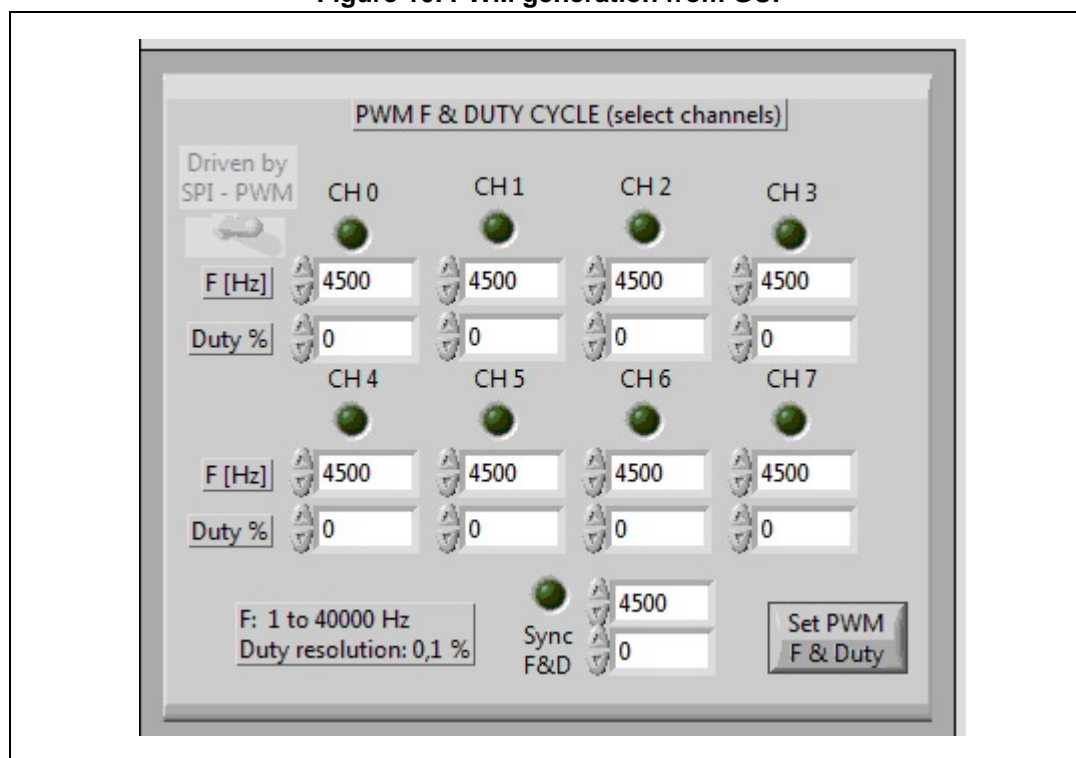


7 PWM driving

After the initial phase of MCU configuration, starting from the "Normal" state of the MCU, it is possible to drive peripherals such as PWM, SPI, SPI PULSE (a specific SPI communication mode ad-hoc conceived for the L9301 device), and so on.

Concerning PWM, frequency (from 1 Hz to 40 kHz with step of 1 Hz) and duty-cycle (from 0 to 100% with step of 1%) can be set for each of the 8 channels of the PWM peripheral. In L9301 application, the maximum value of the frequency is 10 kHz. For more details one can refer to [Figure 10](#).

Figure 10. PWM generation from GUI

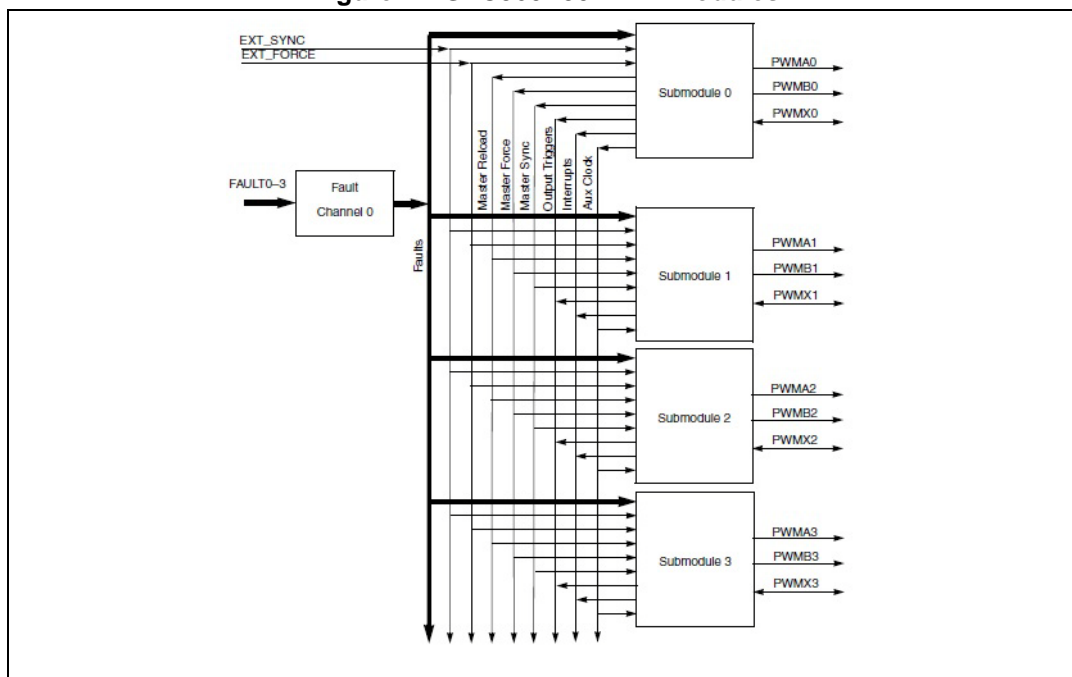


The 8 PWM channels on SPC560P50 MCU have been selected according to the following steps.

First of all, on SPC560P50 MCU there are 12 PWM generated by 4 submodules (0, 1, 2, 3). This induces that 3 channel groups have the same frequency values, nevertheless duty-cycle setting is independent. PWM channels can be activated and masked.

All this preliminary information is useful to generate 8 PWM signals for the 4 peripheral submodules on the 2 channels A[0,1,2,3] and B[0,1,2,3]. Please refer to [Figure 11](#).

Figure 11. SPC560P50 PWM modules



The "MASK" register for the A and B channels is a double buffered register, so to change the value and to be updated it needs a double FORCE_OUT at 1 in the CONTROL 2 register of submodule. This induces an implementation of some command lines into the firmware code, named "temporary ST world solution for PWM bug" (see [Figure 12](#)). These commands lines are used to disable the channel MASK.

Figure 12. PWM unlock mask per SPC560P50

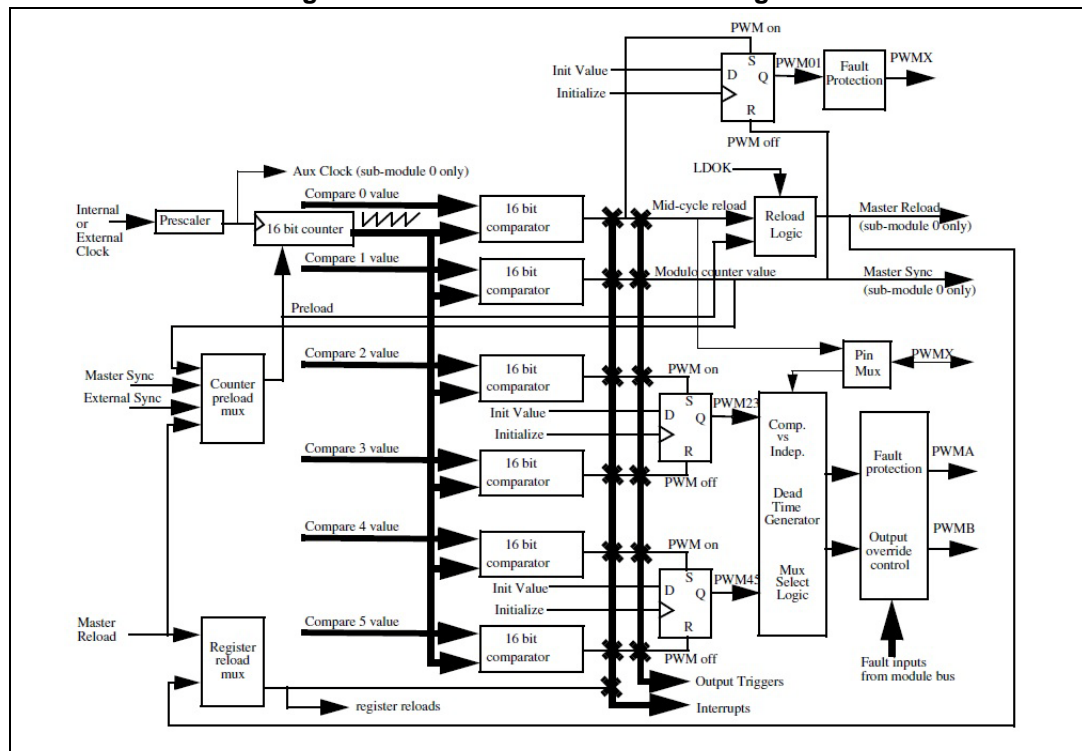
```

FLEXPWM_0.MASK.R = 0x00;
FLEXPWM_0.SUB[addr].CTRL2.B.FRCEN = 1U;
FLEXPWM_0.SUB[addr].CTRL2.B.FORCE = 1U;

```

SPC560P50 PWM generation is tuned by a 16 bit counter. This does not allow to obtain all the frequencies from 1 Hz to 40 kHz with a unique prescaler and induces a more complicated management of the PWM generation. In the upper left side of the SPC560P50 PWM block scheme depicted in [Figure 13](#), there is a prescaler, PSC2, that has as input a frequency coming from another prescaler, that is, a divisor with integers (from 1 to 16) of the clock frequency.

Figure 13. SPC560P50 PWM block diagram

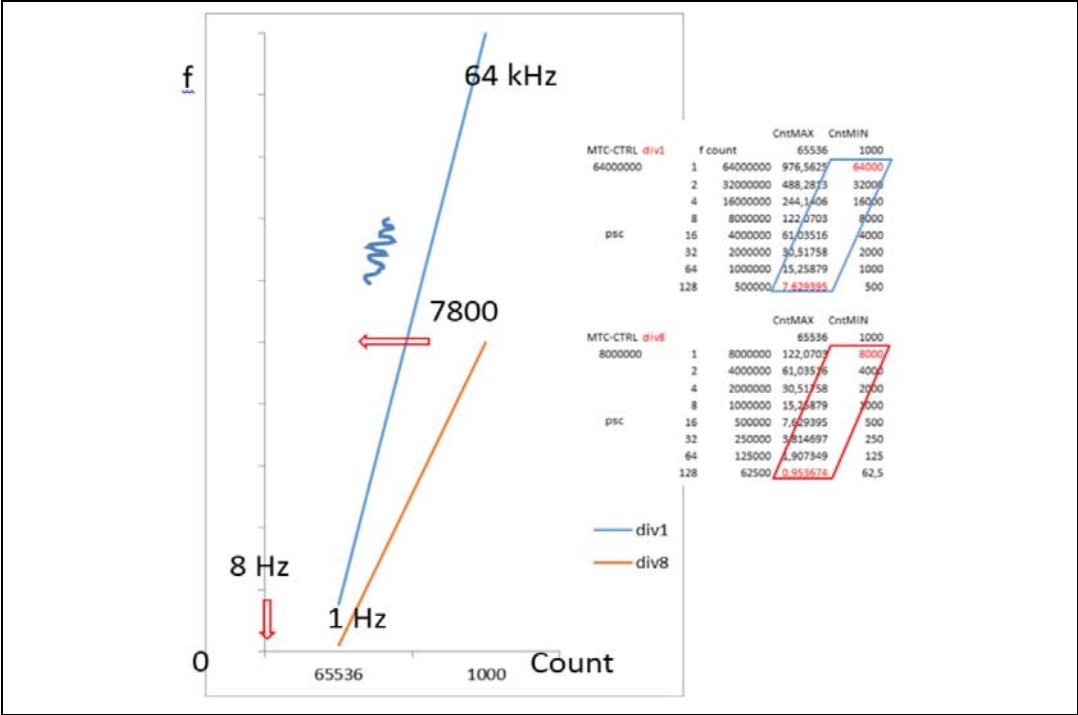


Frequency ranges have been characterized, according to the MTCCTRL div1 and div8:

- with div1, frequencies from 8 Hz to 64 kHz are obtained
- with div8, frequencies from 1 Hz to 7,8 kHz are obtained
- MTCTRL div1 => Clk = 64 MHz / 1
- MTCTRL div8 => Clk = 64 MHz / 8 = 8 MHz

Each change of the MTCTRL is managed with a stop of all the PWM, because the new value could induce the implementation of inaccurate frequencies or not provided for PWM set-up. For both the values of the MTCTRL, the PSC will assume the values 1, 4, 128 in order to maximize the coverage of the frequencies that can be generated and, at the same time, to minimize the change of MTCTRL. Using these values, it has been decided to use an hysteresis to have, under the 8 Hz, a change of MTCTRL to div8. When this change is performed, before setting $f > 7800$ Hz MTCTRL is set to div1.

Figure 14. Frequency hysteresis on SPC560P50 PWM generation



Doing so, it is impossible to set simultaneously 1 Hz and 64 kHz but it allows to obtain continuity in the frequency generation into the range 1 Hz – 7800 Hz and 8 Hz – 64 kHz.

8 SPI driving

Regarding the SPI it is enough to know that SPI is a communication serial protocol between the MCU and the device that uses a signal named "chip select" to enable the communication with a selected device. When the "chip select" is enabled (normally from a high to low transition) the clock of the SPI communication and the related data transmission starts. It can be done when the single SPI send/receive bursts. For this firmware a special function has been designed, named "SPI PULSE", that allows us to send a train of SPI pulse in a time interval with a precision of 10 μ s. This function allows us to drive the channel in frequency and duty-cycle. The "SPI PULSE" function has been conceived to switch-on and switch-off in frequency the channels of an ABS or ESP system by SPI communication. Since MCU locks the unique SPI communication channel, it has been decided to table in a 3-frames structure, the following instructions:

- a first frame for the switch (ON/OFF) of the channels
- a second frame for the complementary switch of the channels
- a third frame, coupled to the second frame, that changes at each 3-frame send (this frame is taken in sequence from a 25 rows diagnostic table)

The first two frames are set from GUI. The third is obtained from a table that contains all the frames for diagnostic task (read and write of the 12 channels plus the device status general register).

At last, there are 27 commands: switch 1 command, switch 2 commands and 25 commands for diagnostic purposes. From the half of the table there are instructions to do the register clear only when there are not faults recognized into the bit gf and got (see [Figure 15](#)).

Figure 15. Frame L9301 MOSI & MISO

MOSI															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W/R		ADD						RES		DATA[19..13]					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[12..0]													CRC[2..0]		

W/R 1 = Write
 0 = Read
 ADD Address
 RES Reserved
 DATA Data
 CRC CRC: Polynomial is x^3+x^2+x+1

MISO															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPIErr[2..0]			CHExcp[5..0]						DATA[19..13]						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[12..0]													CRC[2..0]		

SPIErr[2] SPI Short Frame: a less-than-32bit SPI frame was received
 SPIErr[1] SPI Long Frame: a more-than-32bit SPI frame was received
 SPIErr[0] SPI CRC Error: SPI CRC checksum error
 CHExcp[5] gf: bit 6 of General device diagnostic register
 CHExcp[4] got : bit 4 of General device diagnostic register
 CHExcp[3] vdduv: bit 3 of General device diagnostic register
 CHExcp[2] vddov: bit 2 of General device diagnostic register
 CHExcp[1] vbuv: bit 1 of General device diagnostic register
 CHExcp[0] vbov: bit 0 of General device diagnostic register
 DATA Data
 CRC CRC: Polynomial is x^3+x^2+x+1

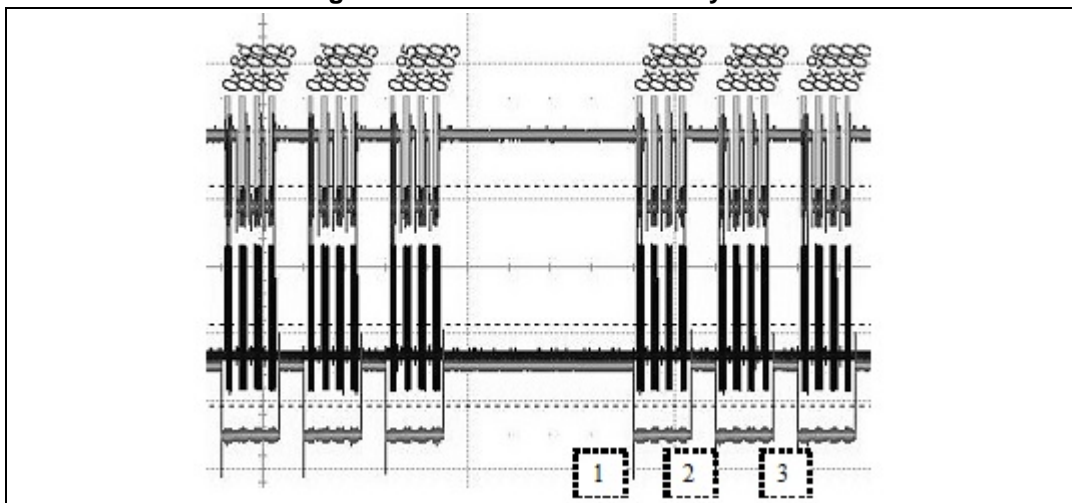
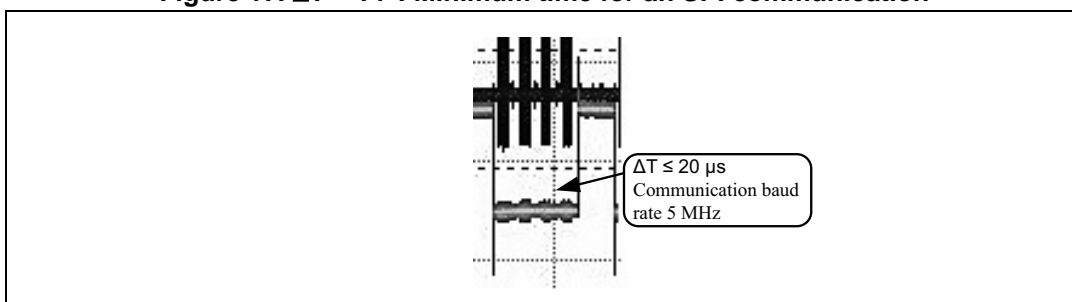
“Switch channel” means an ON/OFF transition of the channels. It can be started by the ‘0’ state (OFF) and changed to ‘1’ state (ON) and vice versa. This builds up the sequence of 32x3 bit, as in [Figure 16](#). In this example, from the bottom to the top of the figure, there are: Chip Select, Clock and MOSI of 32-bit.

For the MOSI signal, one can observe 4 hexadecimal pieces, for instance 0x8D, 0x00, 0x00, 0x05, that is, the 32 bit SPI string "8D000005" in hexadecimal:

- SPI 1 switch channels in a time $T1^\circ$
- SPI 2 switch channels in a time interval equal to ΔT (see [Figure 17](#))
- SPI 3 diagnostic in a time $T2^\circ$

where ΔT is the minimum time to send an SPI pulse, it corresponds to the $T1^\circ$ value plus the half of the $T2^\circ$ value. The time change is done in order to determine the limits of the frequency and duty-cycle.

Figure 16. SPI PULSE 100us duty 20%

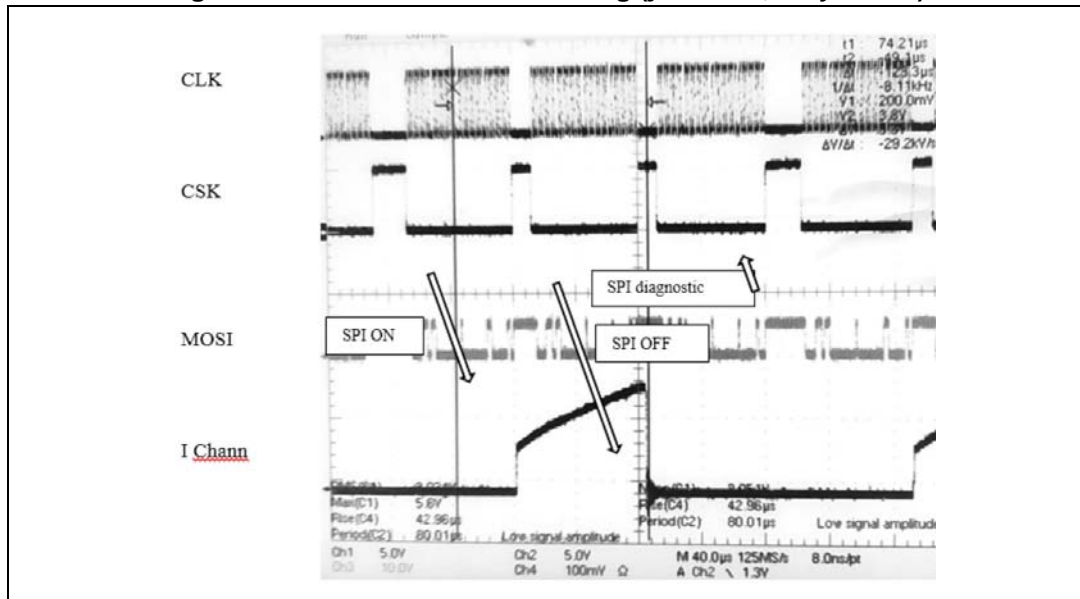
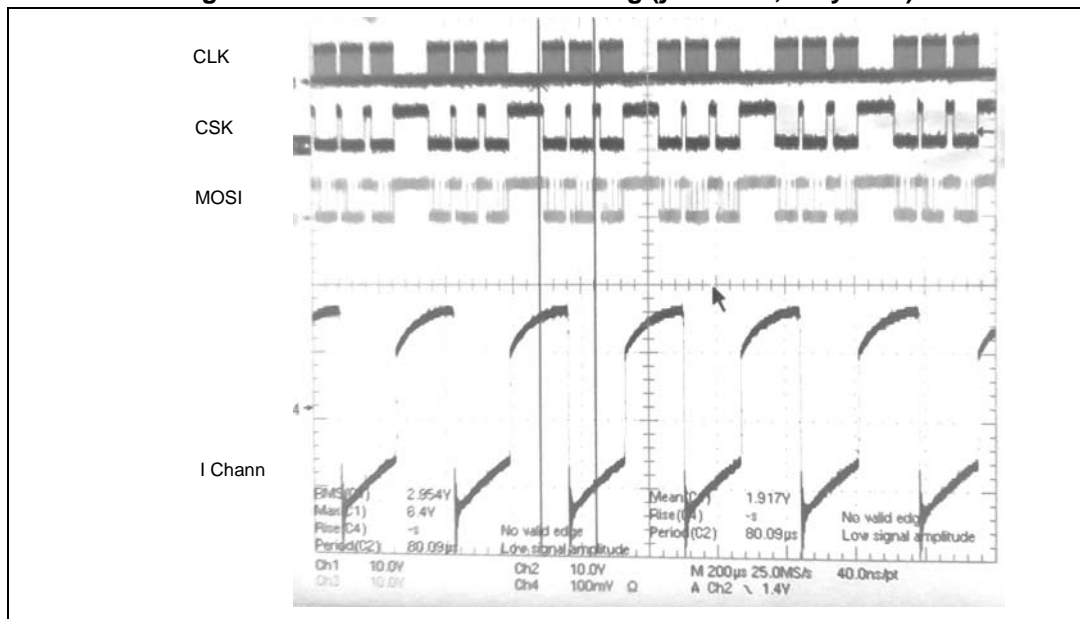
Figure 17. $\Delta T = T1^\circ$: Minimum time for an SPI communication

According to the baud rate, there is the feature of the switch-on and switch-off in frequency of the SPI PULSE. For instance, a baud rate = 500 kHz induces to send a 32-bit SPI, at $\Delta T = 80$.

As a consequence, $f_{\max} = 1/T$ is determined by the three consecutive SPI, with a period = 240 μs , that is, $f_{\max} = 4166$ Hz. If $f = 3000$ Hz, we have $\text{duty}_{\min} = T1^\circ / \Delta T = 80/333 = 24\%$ and $\text{duty}_{\max} = 1 - (T2^\circ/T) = 1 - (160/333) = 52\%$. The duty cycle is not symmetric because frames are odd (in fact there are three frames: switch 1 frame, switch 2 frame and diagnostic frame). To increase the duty_{\max} (with a duty cycle greater than 50%) a switch of the SPI 1 and 2 is induced. Doing so, duty cycle changes from 52% to 76%. When $f > 3000$ Hz, duty_{\max} is less than 50% so there is a discontinuity. For instance $f = 4000$ Hz implies $\text{duty}_{\min} = 32\%$ and $\text{duty}_{\max} = 36\%$; in this case, a switch of the times and SPI frames means to have a duty cycle within range [32–36%] (for duty_{\min}) and [64–68%] (for duty_{\max}).

Below some figures describing an example of driving of a channel of L9301 by SPI PULSE function. When CHIP SELECT is low, the SPI communication starts. For each edge of the CLK signal a bit, composing the frame of the MOSI, is sent from MCU to the L9301 device. There are three MOSI frames: the first frame is for the switch 1, the second frame is for the switch 2, and the third frame is for diagnostic purpose (see [Figure 18](#) and [Figure 19](#)).

In [Figure 18](#), one can observe that the channel is switched-on or switched-off just after the arrival (to the SPI slave device, that is, L9301) of all the 32 bits. The complete SPI sequence is: switch 1 frame — pause — switch 2 frame — diagnostic-frame — switch 1 frame, and so on.

Figure 18. SPI on – SPI off – SPI diag ($f = 3 \text{ kHz}$, duty = 33%)Figure 19. SPI on – SPI off – SPI diag ($f = 3 \text{ kHz}$, duty 50%)

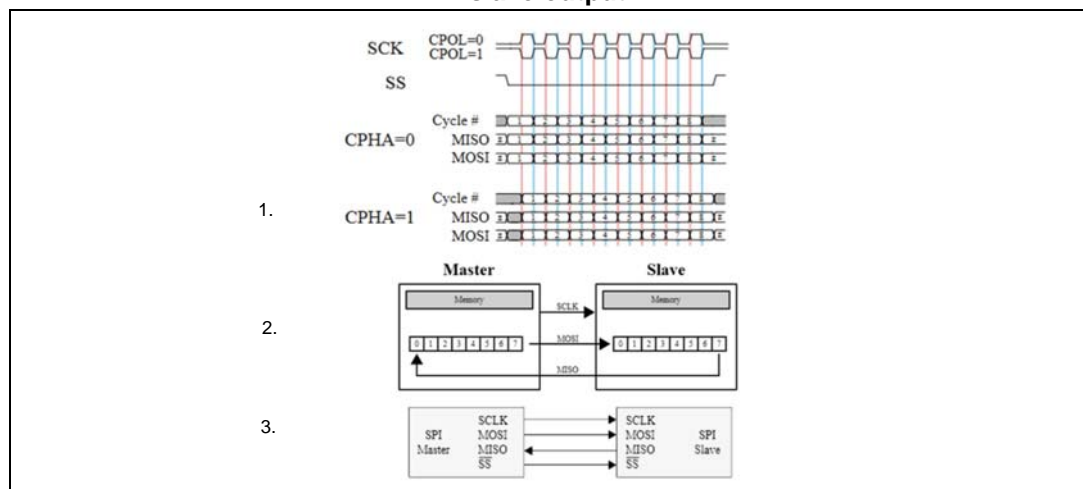
Basically, SPI (Serial Peripheral Interface) is a standard bus for inter-communication among different devices. The data transmission occurs between a master device and one or more slave devices. Master device starts the communication, sends the clock signal and decides when to stop the communication. When the chip select is low, clock signal conventionally starts and for each edge of the clock signal, the master device transmits a bit that is shifted on the slave device and at the same time on the master device an opposite event occurs. For a more detailed description refer to the [Figure 20](#).

There are two other configuration parameters for a SPI communication: CPOL and CPHA.

CPOL is the clock polarity, while CPHA is the clock phase. In order to have a more comprehensive description of the functionality of these configuration parameters user can refer to the following:

- CPOL = 0, clk normally to 0
- CPOL = 1, clk normally to 1
- CPHA = 0, SPI captures/codes the data on the first edge of the clock signal and changes on the second edge
- CPHA = 1, SPI changes data on the first edge of the clock signal and captures/codes on the second edge

Figure 20. SPI communication: SCK or SCLK system clock, SS chip select, CPHA clock phase, CPOL clock polarity, MOSI master output slave input, MISO master input slave output

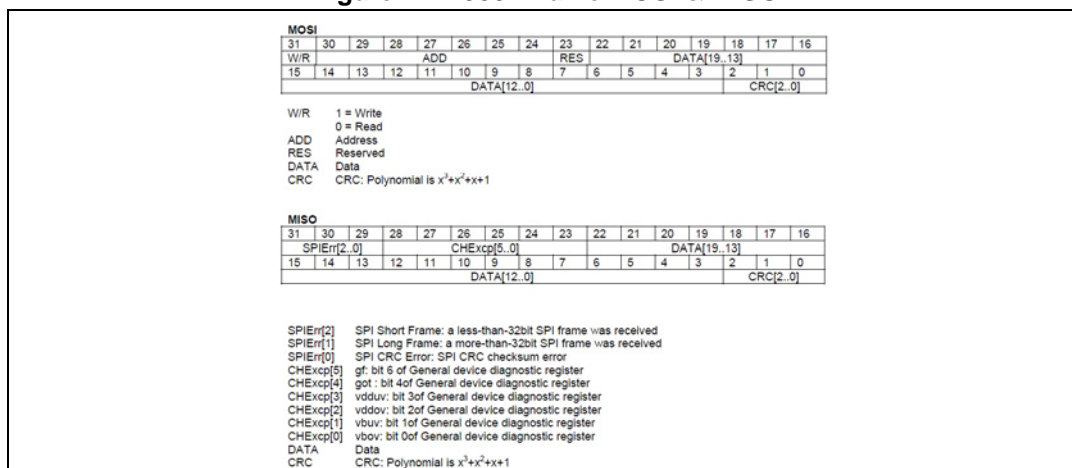


32 bit SPI frames are called MOSI (Master Output / Slave Input) when there is a data transfer from MCU to the L9301 device and MISO (Master Input Slave Output) when there is a data transfer from L9301 to the MCU. MOSI is composed of:

- 1 bit W/R for reading and writing mode
- 7 bit for the ADD address of the registers
- 20 bit for the DATA
- 3 bit for the CRC calculated by means of a three grade polynomial

From a diagnostic point of view, that is, status register 0, 1...12, etc..., if the W/R bit is set a "clear" of the register is driven. Other registers are just read-only registers, so the W/R bit is blocked to 0.

Figure 21. L9301 frame MOSI & MISO



MISO signal received by MCU is composed as follows:

- The last bits 31, 30, 29 are dedicated to the SPI error
- 6 bits are used for diagnostic purposes; there are some bits of the SR0 register (general diagnostic register), in particular CHExcp[5 and 4] indicate faults on the 12 channels of the device (L9301)
- 20 bits are used for DATA
- 3 bits are used for CRC as for MOSI signal

MISO does not contain ADD the address of the registers, so the reading occurs just on the answer after the sent request.

CRC is a control on the compliance of the transmitted data to the rules to be used to code the SPI transmission. It is a kind of parity check and it is performed by a bit to bit XOR shifted towards right of the 32 bits. The polynomial used to code the CRC is x^3+x^2+x+1 , that is, the binary sequence "1111".

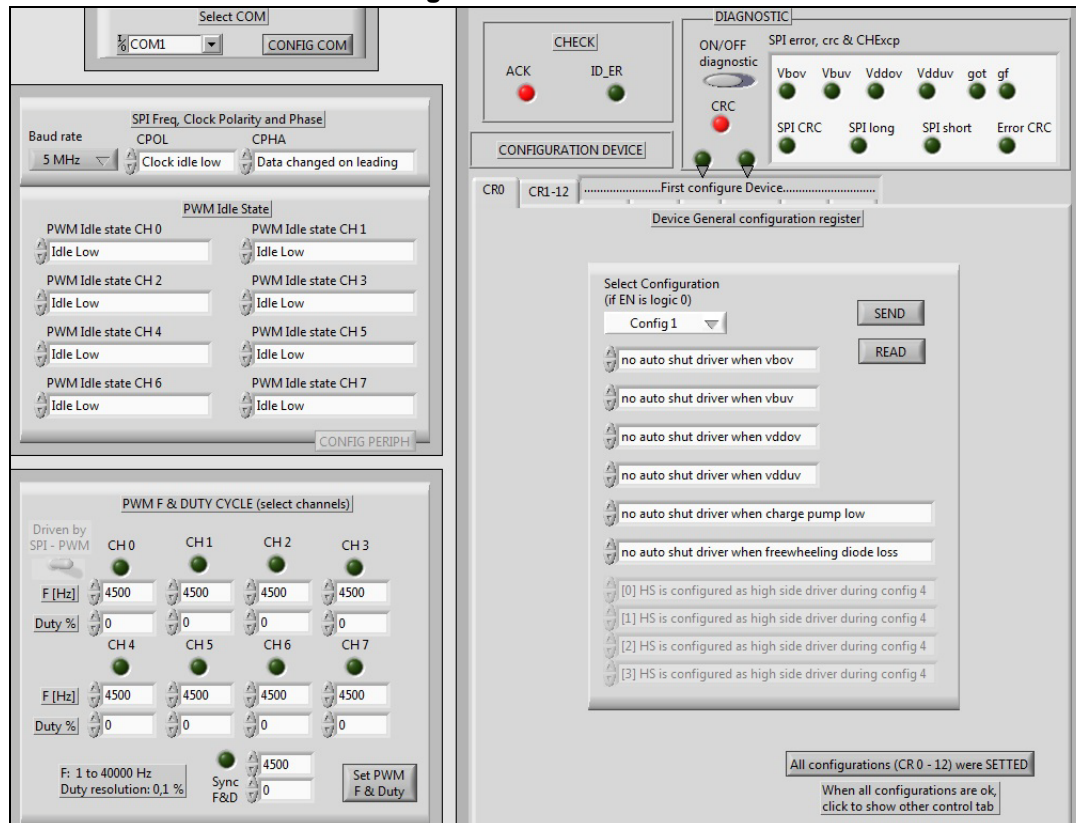
9 GUI general user interface labview

This section is dedicated to the specifications and the mode of working of the device (L9301) by means of a customer-oriented GUI (General User Interface) Labview. GUI has been defined on a basis of configurable and readable bits corresponding to the registers of the device. A preliminary GUI description is shown in the [Figure 22](#). There are two macroscopic parts:

- on the left you can find the configuration of the SPI communication between MCU and device, and the configuration of the PWM driving
- on the right there are all the registers of the device (L9301) useful for the configuration, the driving and the diagnostic purposes

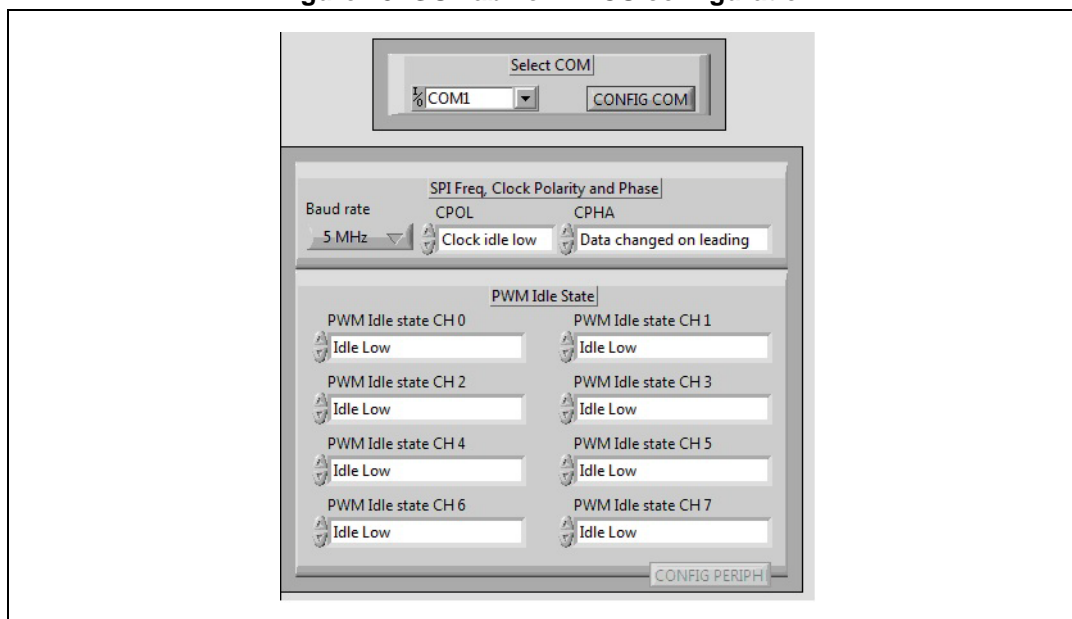
The possibility to have disabled commands on the GUI allow the users to not commit errors during the sequence of the SPI communication.

Figure 22. GUI Labview



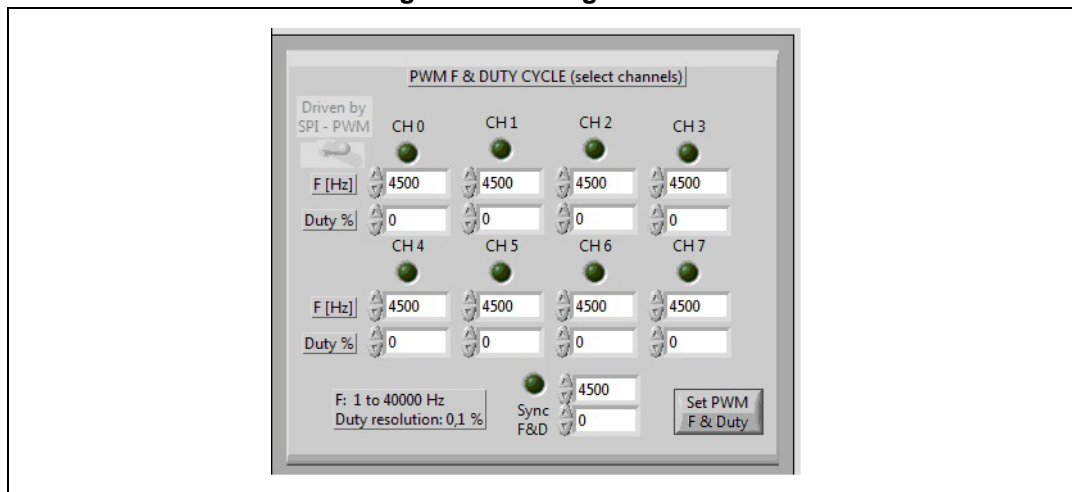
[Figure 23](#) describes the details of the configuration regarding the link between MCU and device L9301.

Figure 23. GUI labview: MCU configuration



GUI part depicted in [Figure 23](#) is the only one enabled when the GUI starts. In this case user can select the COM port for the PC host and MCU communication. After this phase, using the button "config com" it is possible to configure MCU peripherals. In fact the button "config periph" is enabled. After these preliminary phases, user can set SPI parameters and the idle state (if normally low or high) of the 8 PWM channels of L9301 device.

Figure 24. PWM generation



In [Figure 24](#) it is shown how to set the PWM. User can switch-on/switch-off independently each PWM channel by checking the led located under the channel label (CHx). Furthermore, user can choose frequency and duty cycle. Duty cycle will be different to zero only for the enabled channels.

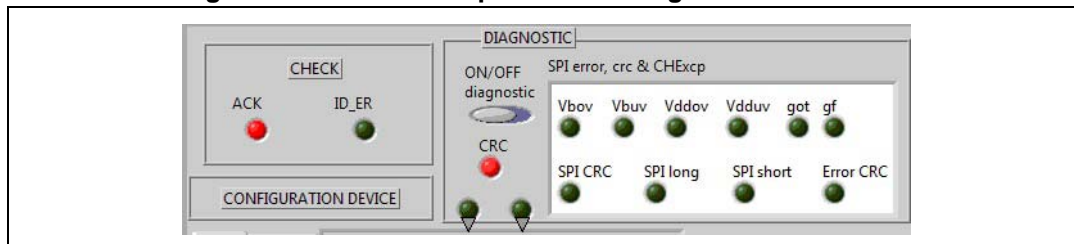
Sync F&D button allows to synchronize more channels on a single couple of (f , duty-cycle) values. When user pushes "Set PWM f & duty" button, PWM is enabled on the selected

channel. Below the range of available values for frequencies and duty-cycle for the PWM generation (MCU side):

- f : [1;40000] Hz, with step of 1 Hz
- duty cycle: [0–100]%, with an accuracy of 0,1%

Since the system level specification of the L9301, from application point of view, forecasts operating condition in ABS/ESP mission profiles up to 10 kHz of driving frequency, this value is the maximum frequency (f_{max}) usable for the power driver (L9301) of the ABS/ESP control unit.

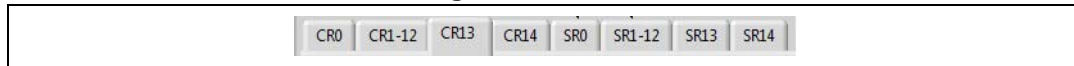
Figure 25. LED used to provide warnings to the GUI users



User can see from [Figure 25](#) there are several windows:

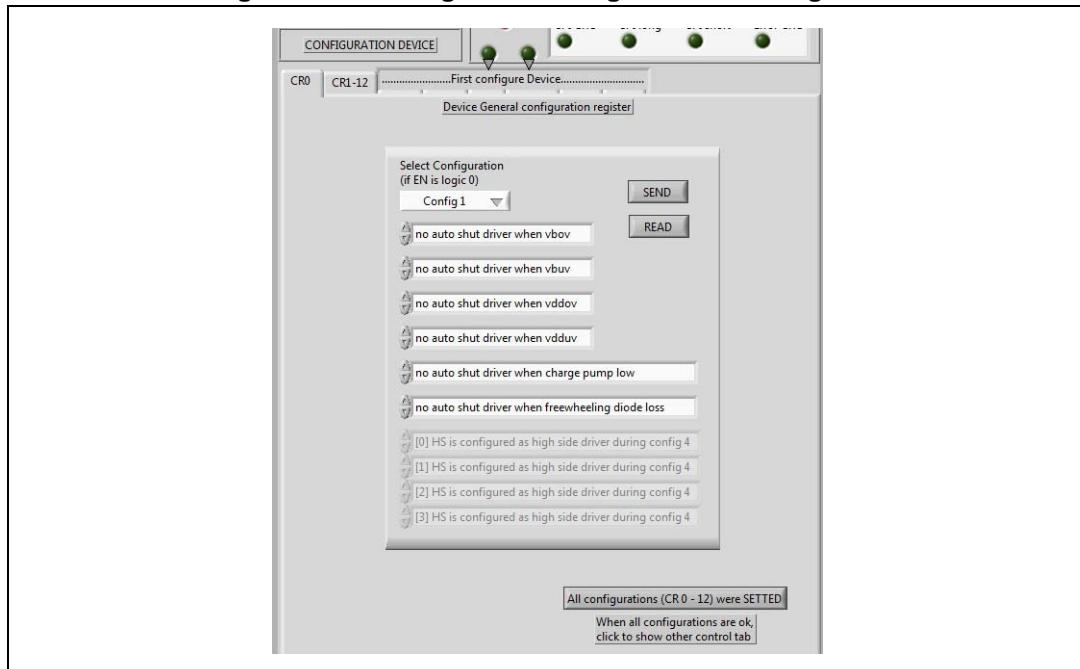
- Check: concerns frame sent to the MCU, that is, if the communication successfully occurred (in this case ACK and ID led are green)
- Configuration device: identifies GUI tabs (see [Figure 26](#)) underlying configuration registers of L9301 device
- Diagnostic: identifies registers underlying diagnostic, switch-on/switch-off of the continuous diagnostic, CRC led of the communication protocol, MISO leds for faults. On the other hand, the leds with a triangle on the bottom indicate a fault occurred on the corresponding register (SRx).

Figure 26. GUI tabs



Follows a detailed description regarding the L9301 functionalities that can be enabled by GUI: configurations, driving, diagnostic. Tab contents, illustrated in the following pages, has been conceived taking into account the structure of the MOSI and MISO registers for the SPI communication between the MCU and the considered device (L9301). Each tab corresponds to a register for a total of 30 tabs. CRx are for the configuration while SRx registers are for the diagnostic.

Figure 27. Device general configuration:CR0 register



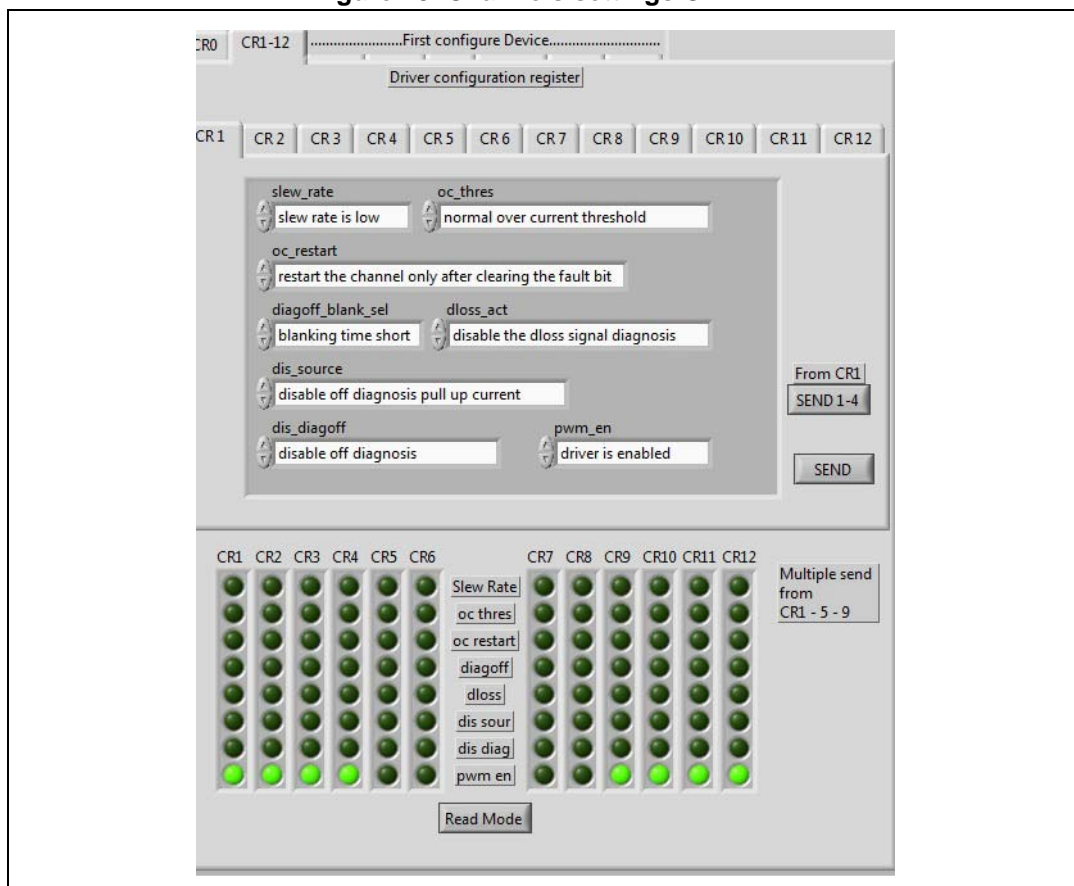
The first tab on the left of [Figure 27](#) is the CR0 register, that is, the general configuration register of the device (L9301). In this tab, it is possible to choose:

- the type of the configuration of the device: from 1 to 4
- to set auto-shut down of the device when different faults occur, for instance, an auto-shut down of the device when the Vdd overvoltage condition is detected etc...

"Send" button is used to transmit the SPI with the selected configuration while "Read" button is used to read the status of the register.

When the registers CR0 –12 have been all set, by pushing the button "All configurations are SETTED", it is possible to enter into the other GUI tabs.

Figure 28. Channels settings CR1–12

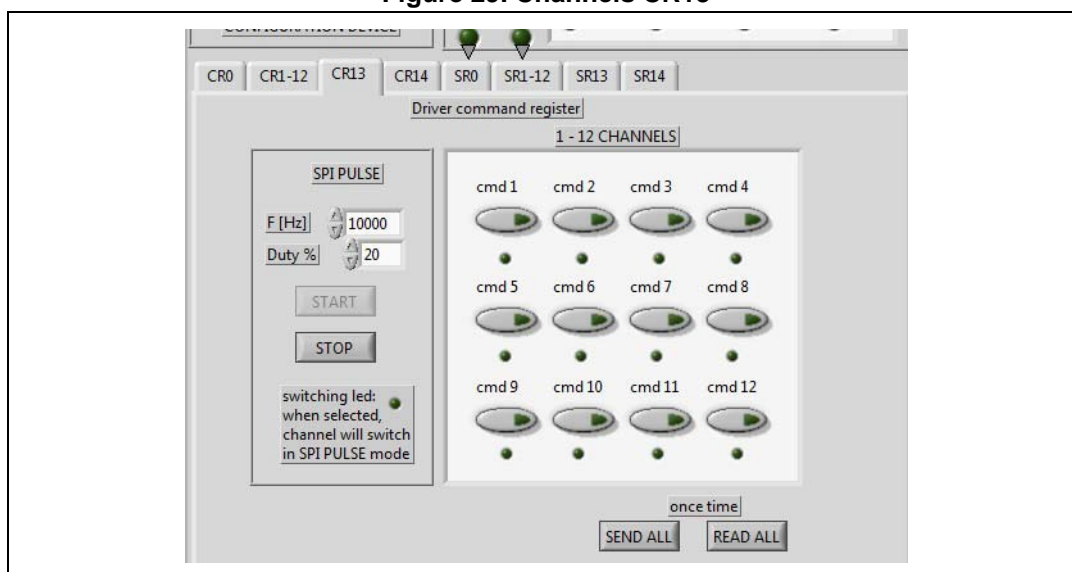


Channel configuration is described in [Figure 28](#). Inside the CR1–12 tab there are 12 other tabs, each one for each channel/register. At the bottom there are leds of all the registers showing simultaneously the status of all the registers. Saving the registers 5 and 8 in which it cannot enable PWM, similar set-up are available for all the channels:

- slew-rate setting-up (high or low)
- overcurrent threshold (normal or double)
- automatic restart of the channel after the fault detection
- blanking time (short or long)
- enable of the diagnosis of the diode loss and pull up current
- channel drive in PWM or SPI

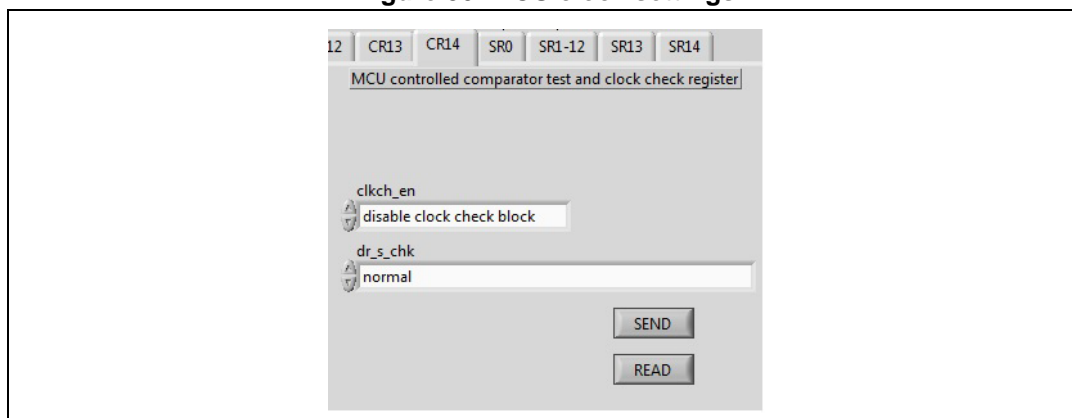
"Read mode" button is available only if SPI PULSE has not been activated by the "Start" command.

Figure 29. Channels CR13



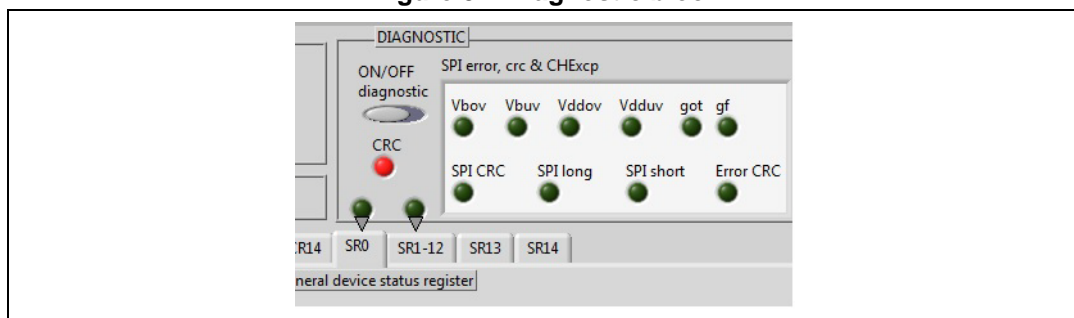
If the selected configuration of the device forecasts the channel driving by SPI, from the tab in [Figure 29](#) it is possible to do the static switch-on/switch-off of the channel by means of "cmdx" button and also pushing the button "Send all". "SPI PULSE" box allows to drive the switch-on/switch-off in frequency of the channel by SPI setting respectively f_e duty cycle. Moreover, "cmdx" leds indicate the initial state of the channels (that is channel starts turned-on or turned-off) and the underlying led, named "switching led", allows to choose to change the state (ON/OFF), otherwise it remains fixed. In SPI PULSE MODE all the part regarding the configuration of the MCU is disabled. On the other hand, it is possible to set PWM and only to read diagnostic ("clearing" of the registers is enabled only when SPI PULSE is blocked).

Figure 30. MCU clock settings



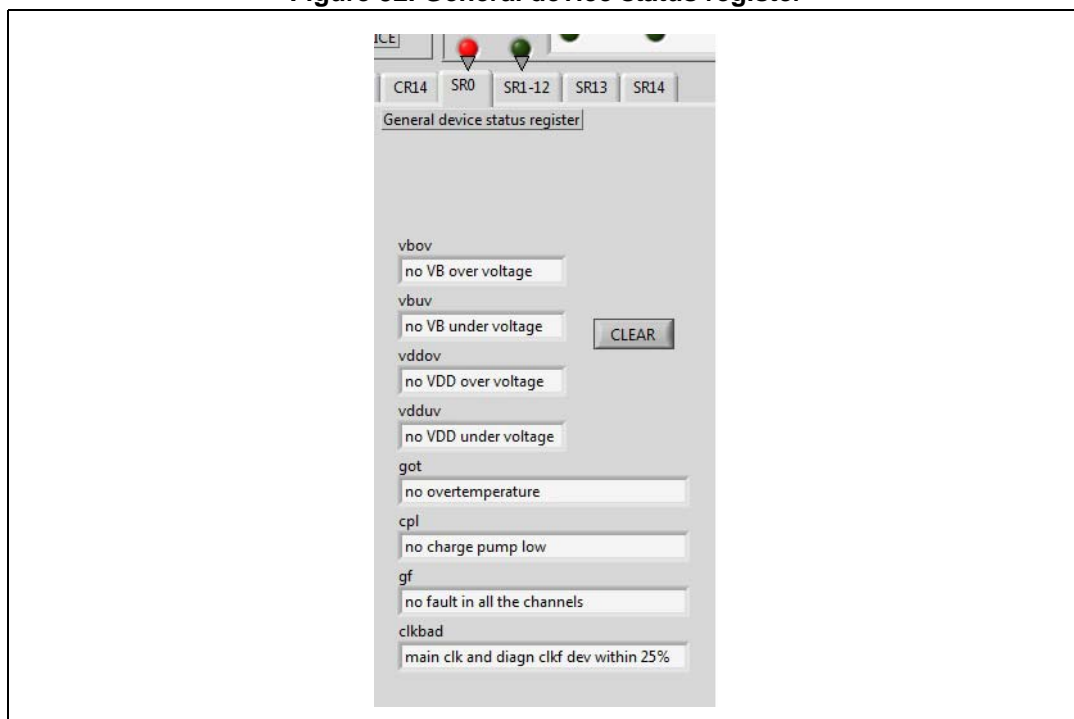
CR14 tab in [Figure 30](#) allows to manage options on the MCU clock: enable or disable the clock check block and input inverting.

Figure 31. Diagnostic block



In the panel described in [Figure 31](#) there is the button to turn-on/turn-off the diagnostic. CRC led highlights issues into the frame from the viewpoint of the CRC check. More in details, in the "SPI error, crc & CHExcp", user can see different leds useful to indicate some faults regarding the SPI communication between MCU and L9301 device (for a more comprehensive meaning user can refer to L9301 datasheet). When "ON/OFF diagnostic" is switched-off, SR0–12 registers are automatically read when the corresponding tabs are selected. Take into account that host PC periodically wants a refresh of the register reading. To do this user can use the "Clear" button. Clearing can be done only manually and only when SPI PULSE is blocked.

Figure 32. General device status register

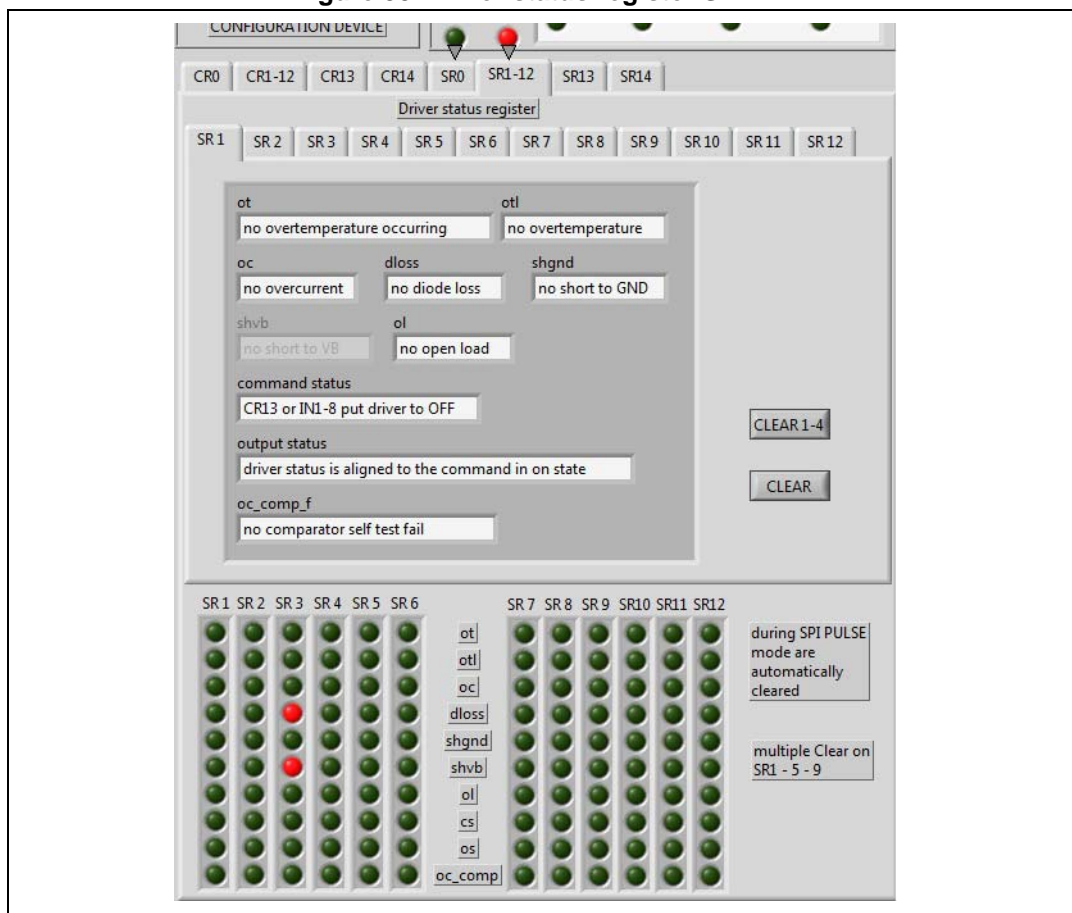


In [Figure 32](#) there is a picture describing the general device (L9301) status register. In this register, user can see the main faults of the device. In the considered example, led in red light indicates the SR0 tab in order to show that there is a fault that has to be read. SR0 register can highlight if there is a fault related to an overvoltage or undervoltage condition on Vb or Vdd. Furthermore, in SR0 register user can find also got and gf bits, where:

- got: general over temperature, is a logic OR of all the channels and indicates if there is an overtemperature on at least a channel of the device
- gf: general fault, is a logic OR of all the main issues relating to the operating conditions of the device channels

In other words, these two bits (that are got and gf) show if there are some issues on the device channels (all the 12 channels) and this information is available on each received MISO frame. Finally, got and gf are the bits in only reading and they can be reset if and only if the issue underlying the fault is solved.

Figure 33. Driver status register SR1–12



In [Figure 33](#) user can observe the details of the GUI regarding diagnostic operations on all the 12 channels of L9301 device. There are some tabs similar to the tabs used for the setting-up operations described before. Red led at the top of the figure indicates a fault on one of these registers. "cs" led linked to "command status" is not a fault but indicates if the channel is turned-on or turned-off. At the bottom of the figure there are leds of all the 12 channels in order to have a view on the fault occurring on the device channels. Therefore, it is possible to read the following fault conditions: overtemperature, overcurrent, diode loss, short-to- gnd, short to battery, open load, if the channel is turned-on or turned-off, output status and finally overcurrent comparator self test. As mentioned before, when diagnostic is turned-off, SRx is read and updated automatically when the corresponding tab is selected.

Figure 34. Asic version and BIST

At last, in [Figure 34](#), the last two registers describing the silicon version and the status of BIST (Built In Self Test) control, are highlighted. ASIC writes bit equal to 1 if a fault in the status registers occurs and it cannot be cleared as long as the corresponding register is not activated.

BIST is an internal feature that allows the device to test itself. At the start-up of L9301 device, in the first 11 ms a self-test of the device is performed in order to control if there are some issues related to the device build process.

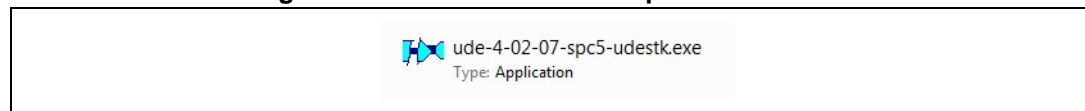
10 SPC560P–DISP: USB drivers installation

The GUI needs a dedicated USB driver to enable the serial communication channel. If the driver is not yet installed or not appropriately configured, the following procedure describes how to install the Driver or to update the current version.

In order to install the drivers, the board SPC560P–DISP does not need the external supplies; the USB connection provides the supply voltage.

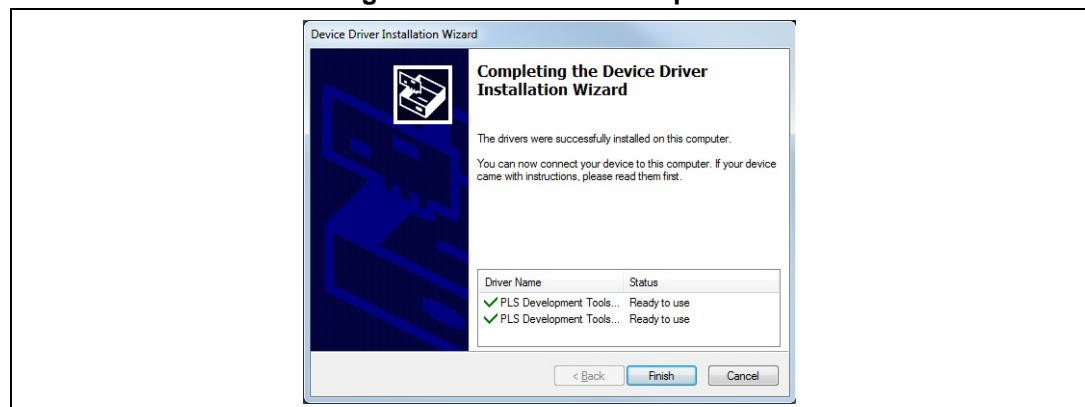
1. Disconnect the USB from SPC560P–Discovery
2. If the UDE Visual Platform 4.2 is already installed as well as the drivers are updated go to step #7 otherwise continue to next step
3. Download the SPC5–UDE/STK 4.02.07 from the link: pls-mc.com/downloads-a-682.html
4. Install the software, right click on the icon named "ude-4-02-07-spc5-udestk.exe" as shown in [Figure 35](#), then select "Run as administrator"

Figure 35. Icon ude-4-02-07-spc5-udestk.exe



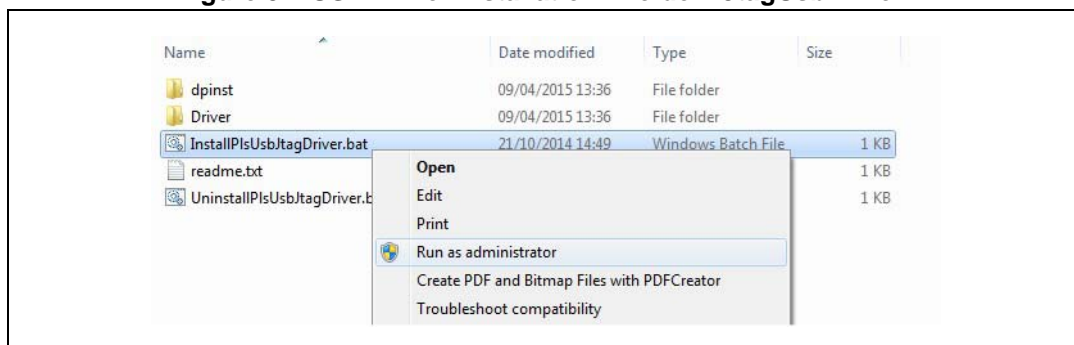
5. Accept to install the USB drivers
6. The drivers installation is completed when the window here below appears (see [Figure 36](#)). When pressing the "Finish" button the installation will be completed and the installation program will be closed

Figure 36. Installation completion

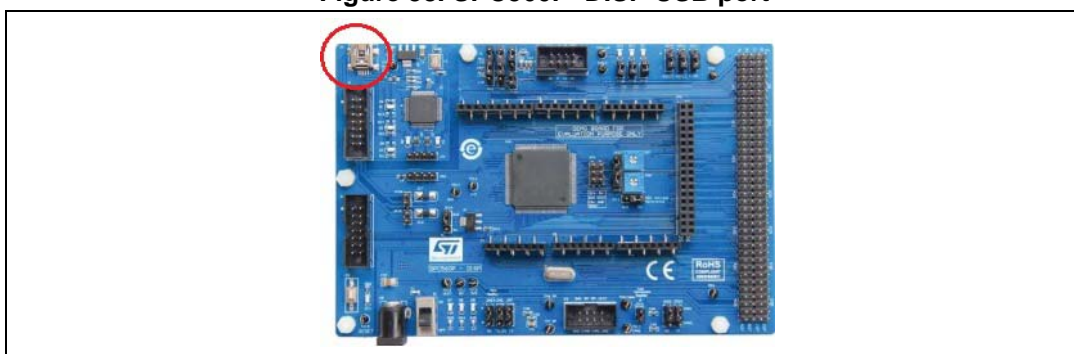


7. Open the folder "C:\Program Files (x86)\pls\UDE 4.2\Driver\JtagUsbDriver"^(a)
8. Right click on "InstallPLsUsbJtagDriver.bat" then select "Run as administrator" as shown in [Figure 37](#) below

a. The path could be different because it depends on the choice the user made during the installation procedure

Figure 37. USB Driver Installation – folder "JtagUsbDriver"

9. Once the installation is completed, connect the USB cable to the SPC560P-DISP board. See [Figure 38](#)

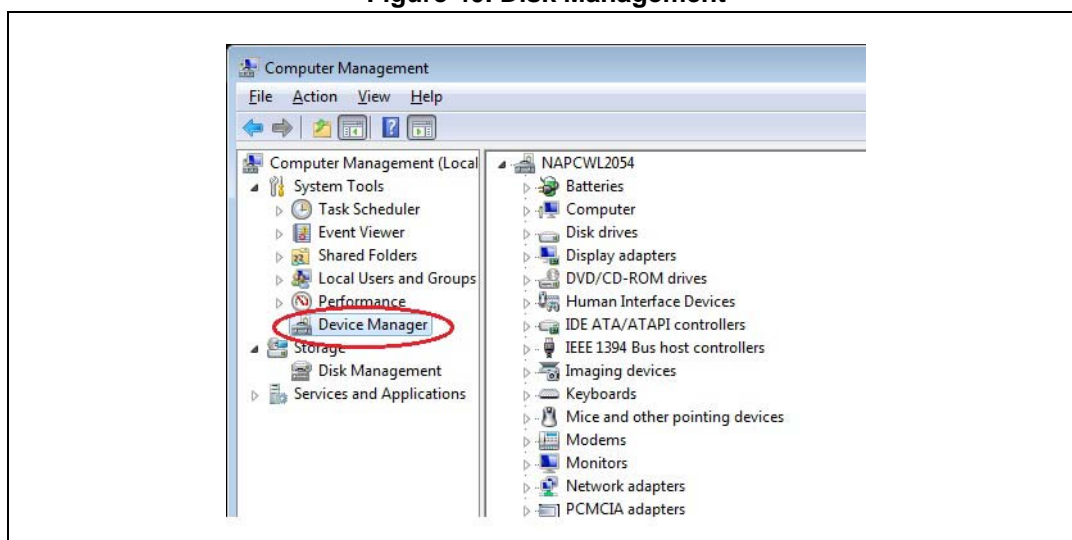
Figure 38. SPC560P-DISP USB port

10. From "Start" menu, right click on "Computer" item then select "Manage" as shown in [Figure 39](#)

Figure 39. Start menu/Computer/Manage

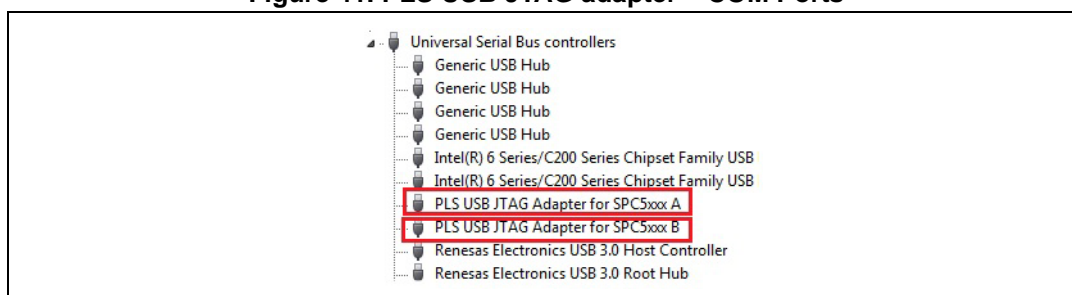
11. Once the Computer management popup appears, select "Device Manager" from the System Tools menu as shown in [Figure 40](#)

Figure 40. Disk Management

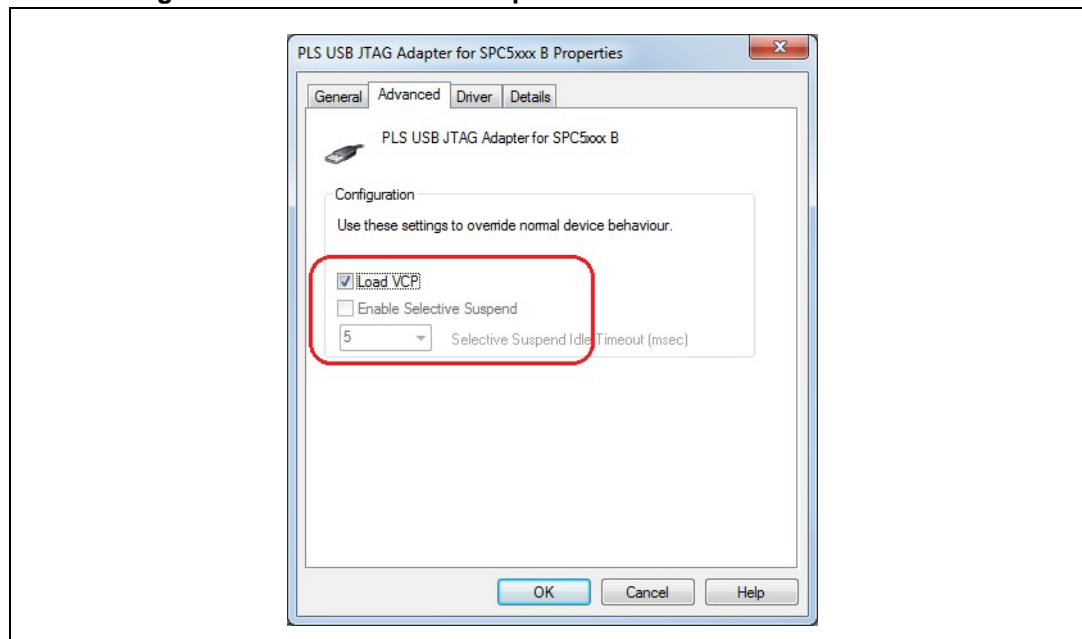


12. Expand the item "Universal Serial Bus controllers", identify "PLS USB JTAG Adapter for SPC5xxx A" and "PLS USB JTAG Adapter for SPC5xxx B" (see [Figure 41](#))

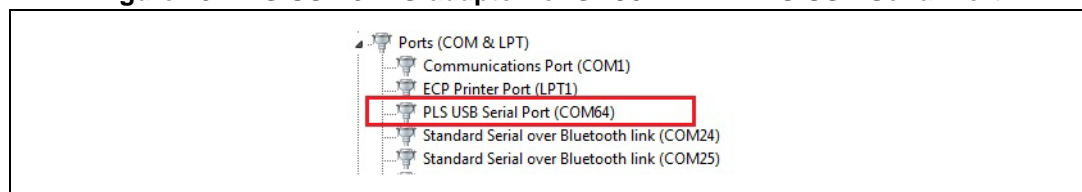
Figure 41. PLS USB JTAG adapter – COM Ports



13. To enable the COM port, right click on "PLS USB JTAG adapter for SPC5xxx B" then click "Properties" and select the "Advanced" tab. Flag the "LOAD VCP" (Virtual COM Port) box as shown in [Figure 42](#).

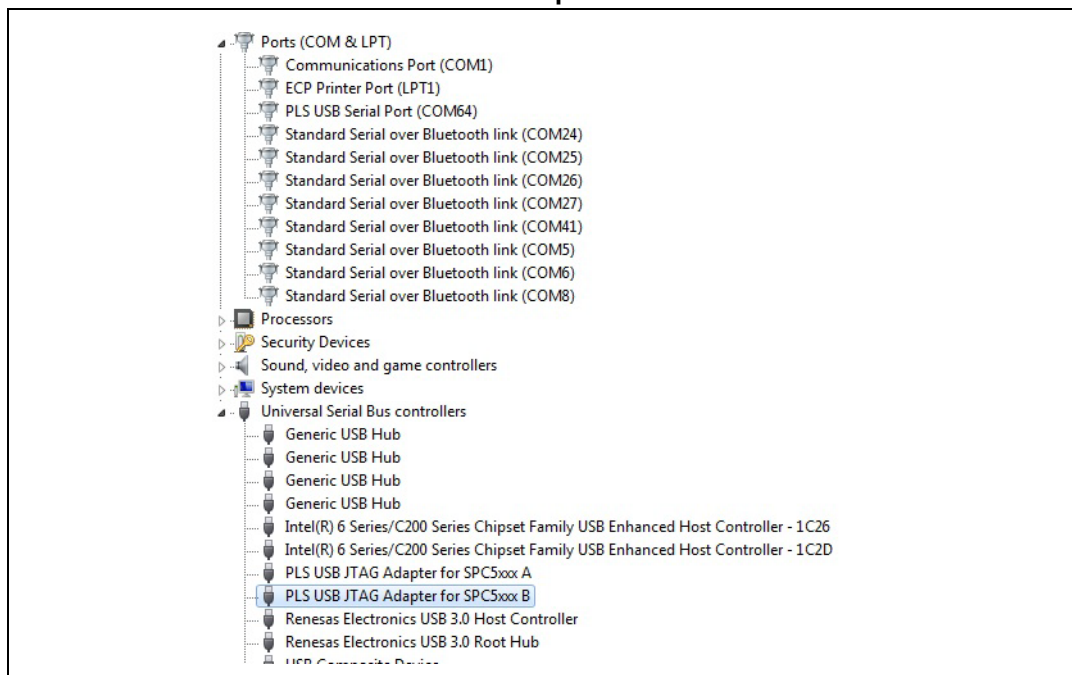
Figure 42. PLS USB JTAG adapter for SPC5xxx B – Load VCP box

14. Disconnect the USB cable from the SPC56P-Discovery then reconnect. A new COM port appears, and Windows will install the new drivers automatically. From the “Device manager” window, check the new COM port available (see [Figure 43](#)).

Figure 43. PLS USB JTAG adapter for SPC5xxx B – PLS USB Serial Port

15. The COM port is configured and available to be used for serial communication with the PC (see [Figure 44](#)).

Figure 44. PLS USB drivers and serial port – Device manager window after installation procedure



11 Bibliography

- [1] "LabVIEW 2012 Core 1", National Instrument, 2012
- [2] "LabVIEW 2012 Core 2", National Instrument, 2012
- [3] "L9301 datasheet", STMicroelectronics internal reference, 2014
- [4] "SPC560P50 Reference Manual", STMicroelectronics, 2013

12 Revision history

Table 2. Document revision history

Date	Revision	Changes
19-Jan-2017	1	Initial release.



IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved