make a figure

Robotics and Electronics

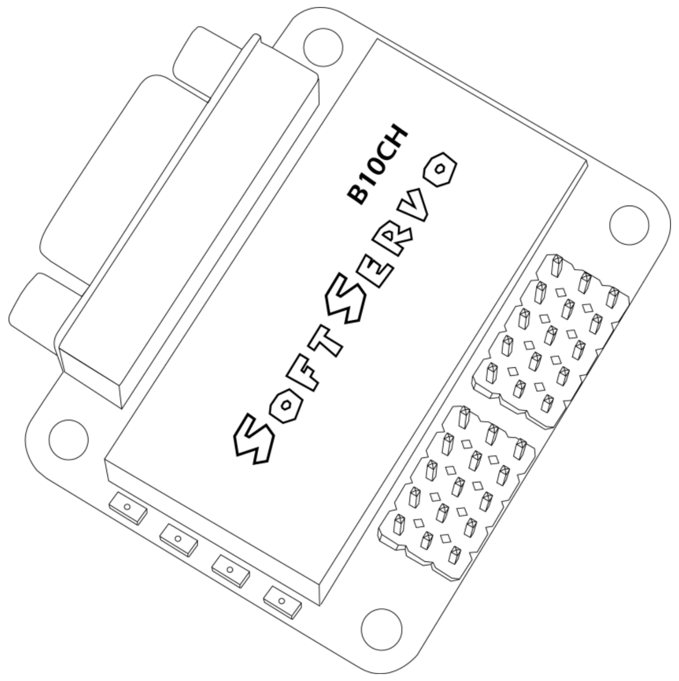
SoftServo-B10CH

Introduction

Strictly speaking, **SoftServo-B10CH** is a 10-channel PWM signals generator, which can be applied to most of the devices that integrate RC interface, but the original intention of the design is to control the *PWM-Style* servo. It integrates RS485 and I2C into one connector, and they share the same user register, so the servo controller can be used via any one of the two interfaces at any time. All functions can be performed by reading or writing the user register, except for a few special operations(please refer to the corresponding communication protocols for details), which is similar to the MCU operation. In addition, the high-level width of the PWM signal is simplified as *PWM-Effective-Width* which is typically from 500μs to 2500μs.

Specification

Supply Voltage	: DC 5 ~ 12V
Baudrate Range	: 1200 ~ 115200bps
I2C Speed	: 100KHz
Size	: 44mm * 47mm * 13mm (L*W*H)



Pin Definition

1. RS485-A(D+)
2. RS485-B(D-)
3. I2C-SDA
4. I2C-SCL
5. GND
6. VCC
7. VCC
8. Error
9. GND

Features

- All the PWM signals output at the same time(in other words, they almost have the same rising edge), and the actual resolution is up to 1μs(the hardware version must be higher than V12).
- The signal coupling is so low that it is difficult to detect visually, just like all the PWM signals are generated by separate hardware.
- It integrates the *Soft-Start* function. As we all know, the initial position of the *PWM-Style* servo is typically unknown. *Conventional-Start* mode makes servo rotate to the goal position with full speed, which is unsatisfactory in most robotic applications. *Soft-Start* mode can make servo rotate slowly when the current position is unknown, but need more current in a short time. It is suggested that all the servos should be started in time-sharing mode to reduce the electric current when you need to control several servos.
- The cycle, the minimum *PWM-Effective-Width* and the maximum *PWM-Effective-Width* can be set flexibly, which makes all the *PWM-Style* servos compatible.
- It is based on a subdivision algorithm, the maximum subdivision value is up to 250. The position and the speed of each channel can be controlled separately, which is conductive to robot dynamic control. The speed depends on the subdivision(see below).
- All the PWM signals can be terminated by setting the equivalent position value to 251 at any time, which is extremely useful for protecting the servo.
- It integrates a stable and high-efficiency RS485, and the baud rate can be detected automatically. There is no instruction packet loss when the baud rate is not higher than 28800bps, otherwise the first instruction packet will be discarded. Furthermore, it is recommended to send the sync bytes(0xff 0xff 0xff 0xff) first when the baud rate is higher than 28800bps.
- It integrates a stable and high-efficiency I2C.

User Register Definition

Address	Name	Store	Access	Initial Value	Description
1(0x01)	ProtocolVer	ROM	R	0x__	Protocol Version (RS485 / I2C)

2(0x02)	DeviceMainClass	ROM	R	0x02	Device Main Class (0x02 : Servo Controller)
3(0x03)	DeviceSubClass	ROM	R	0x01	Device Sub Class (0x01 : SoftServo Series)
4(0x04)	HardwareVer	ROM	R	0x1_	Hardware Version (0x1_ : B10CH)
5(0x05)	SoftwareVer	ROM	R	0x1_	Software Version (0x1_ : B10CH)
6(0x06)	Command	RAM	W	0x00	Command Byte
7(0x07)	ID	ROM	R/W	0x01	Device Address (1~126, 0 : Broadcast ID)
8(0x08)	PWM_Cycle	ROM	R/W	0x14	The cycle of the PWM signal (5~250, x1ms)
9(0x09)	PWM_MinL	ROM	R/W	0xf4	The lower byte of the minimum *PWM-Effective-Width*
10(0x0a)	PWM_MinH	ROM	R/W	0x01	The higher byte of the minimum *PWM-Effective-Width*
11(0x0b)	PWM_MaxL	ROM	R/W	0xc4	The lower byte of the maximum *PWM-Effective-Width*
12(0x0c)	PWM_MaxH	ROM	R/W	0x09	The higher byte of the maximum *PWM-Effective-Width*
13(0x0d)	Position1	RAM	R/W	0xfb	The equivalent position of Channel 1
14(0x0e)	Position2	RAM	R/W	0xfb	The equivalent position of Channel 2
15(0x0f)	Position3	RAM	R/W	0xfb	The equivalent position of Channel 3
16(0x10)	Position4	RAM	R/W	0xfb	The equivalent position of Channel 4
17(0x11)	Position5	RAM	R/W	0xfb	The equivalent position of Channel 5
18(0x12)	Position6	RAM	R/W	0xfb	The equivalent position of Channel 6
19(0x13)	Position7	RAM	R/W	0xfb	The equivalent position of Channel 7
20(0x14)	Position8	RAM	R/W	0xfb	The equivalent position of Channel 8
21(0x15)	Position9	RAM	R/W	0xfb	The equivalent position of Channel 9
22(0x16)	Position10	RAM	R/W	0xfb	The equivalent position of Channel 10
23(0x17)	Subdivision1	RAM	R/W	0x01	The equivalent speed of Channel 1
24(0x18)	Subdivision2	RAM	R/W	0x01	The equivalent speed of Channel 2
25(0x19)	Subdivision3	RAM	R/W	0x01	The equivalent speed of Channel 3
26(0x1a)	Subdivision4	RAM	R/W	0x01	The equivalent speed of Channel 4

27(0x1b)	Subdivision5	RAM	R/W	0x01	The equivalent speed of Channel 5
28(0x1c)	Subdivision6	RAM	R/W	0x01	The equivalent speed of Channel 6
29(0x1d)	Subdivision7	RAM	R/W	0x01	The equivalent speed of Channel 7
30(0x1e)	Subdivision8	RAM	R/W	0x01	The equivalent speed of Channel 8
31(0x1f)	Subdivision9	RAM	R/W	0x01	The equivalent speed of Channel 9
32(0x20)	Subdivision10	RAM	R/W	0x01	The equivalent speed of Channel 10
33(0x21)	CurrentVoltage(1)	RAM	R	0x__	This value is 8 times the current power supply

1: The hardware version must be higher than V11.

Command Byte Definition

What is the role of the command byte plays? Firstly, 100 user registers may not be able to meet the needs of all functions. The command byte is reserved for unexpected needs. Theoretically, there are 255 types of operations can be extended by changing the command byte. Secondly, sometimes it is hoped that some special operations can be performed by changing a command byte with some special values(easier to remember). Note that the software version must be higher than V14 when you set the command byte to “0xf0” or “0xf1”.

- 0xbf : Enable the *Soft-Start* mode.
- 0xbc : Enable the *Conventional-Start* mode(the default start mode after power-on).
- 0xcc : Synchronize the clock which is responsible for generating PWM signal.
- 0xf0 : When the PWM signal output again after termination, the breakpoint of the equivalent position will be used as the reference point.
- 0xf1 : When the PWM signal output again after termination, the middle position will be used as the reference point.

Technical Q & A

- **How to evaluate a PWM-Style servo controller**?

1. The *PWM-Effective-Width* of servo is not always the same. Taking the compatibility into account, it is necessary that the range of the *PWM-Effective-Width* can be set flexibly.
2. For the standard servo, the cycle of the PWM signal is typically 20ms, but a higher frequency PWM signal can be received by the digital servo. When an application system requires a higher response rate, the cycle of the PWM signal needs to be shorten. That is to say, the cycle of the PWM signal should also be able to be set flexibly.
3. It is hoped that all the PWM signals can output at the same time and the signal coupling could not be detected visually.
4. In some dynamic application systems, especially for the humanoid robot, whose dynamic balance is achieved by the inertia force, it is hoped that the position and the speed of each channel can be separately controlled at any time.
5. Most of the servo controllers store the servo action code in the EEPROM or Flash ROM, perhaps, it is simpler for operation. In our opinion, the servo action code of a robotic application system is better to be dynamically generated combining the sensors and intelligent algorithms. In addition, some features of the communication also need to be evaluated, such as the simplicity, the real-time capability and the stability.
6. In order to protect the servo, the PWM signal must be able to be terminated at any time. It is often seen that servo is damaged by the impact force. Theoretically, if there is any abnormal impact force, the PWM signal must be terminated. For example, once the robot is in the falling state, which can be detected by the accelerometer sensor, if the PWM signal can be immediately terminated to make the servo working in the coast or brake state, the rigid impact force would be avoided.

- **What is the mapping between the PWM-Effective-Width, the equivalent position and the actual position?**

If the *PWM-Effective-Width* is directly used as a parameter of an instruction packet, each position needs two bytes to describe. The theoretical position-controlled resolution can be improved to the microsecond level, however, it would greatly increase the length of the instruction packet. Taking into account the trade-off between the position-controlled resolution and the real-time capability of the communication, what stated below seems to be a reasonable solution. In addition, the PWM signal can be terminated when the equivalent position value is set to 251, and it is strongly recommended to use the *PWM-Style* servo which is locked by the periodic PWM signal(not a single PWM signal).

The variation of the *PWM-Effective-Width* is always divided into 250 parts. For example, the *PWM-Effective-Width* is from 500μs to 2500μs, and the corresponding angle is from 0° to 180°. The mapping is as follows (“X” is from 0 to 250).

PWM-Effective-Width (μs)	Equivalent Position	Actual Position (°)
500	0	0
2500	250	180
(2500-500)*X/250+500	X	180*X/250
Low-Level	251	Coast, Brake or Lock (according to the servo)

- **What is the subdivision in the servo controller**

The subdivision is the equivalent speed. In short, the speed can be controlled by the subdivision. Furthermore, the subdivision value is inversely proportional to the speed. For example, if the *PWM-Effective-Width* is from 500μs to 2500μs, the subdivision value is 240, the start equivalent position value is 30 and the goal equivalent position value is 60. The variation of the pulse width is equal to “(2500-500)*(60-30)/250=240μs”, which would be completed within 240 PWM signal cycles.

- **What is the difference between the Bulk-Transmission and the Control-Transmission?**

It is simply mentioned in the RS485 communication protocol. In the servo applications, sometimes we focus on the precision of the pulse width, while we focus on the stability of the communication(any of a valid instruction packet could not be discarded) at other times. Although they are obviously contradictory, the contradiction can become less obvious by improving the processing speed of the MCU and optimizing the code. Taking into account the compatibility of the signal level and the simplicity of the circuit board, we prefer the MCU with operating voltage of 5V and make the above-mentioned contradiction become less obvious by optimizing the code. Of course, all the incoming data can be received by DMA, and the above-mentioned contradiction can be resolved completely, however, most of the MCUs that integrate DMA function work at 3.3V or below. Because the clock of the I2C can be stretched, the *Bulk-Transmission* and *Control-Transmission* are only for the RS485(Note: in the *Bulk-Transmission* mode, the baud rate must be higher than 10000bps). Perhaps you still wonder that how to ensure an instruction packet could be correctly received in the *Bulk-Transmission* mode. It is an effective solution that an instruction packet would be sent three or more times(3 is recommended).

- **How to restore factory settings?**

SoftServo-B10CH can be restored to the factory settings by sending an instruction packet via RS485 or I2C at any time, in addition, a set of hardware operations(see bellow) can also complete the restore operation.

Hardware Operations:

1. Cut off the power supply completely.
2. Short circuit between the two pad holes which as shown in the picture on the right side, and please Keep the step 2 until the step 3 finished.
3. LED-Error, LED-Rx and LED-Tx will go on simultaneously after power-on.