SR518 Robot Servo Manual

1.Components In Brief

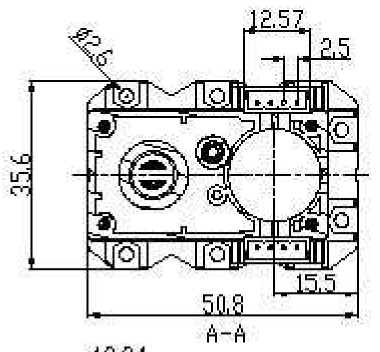
1.1 Product characteristics

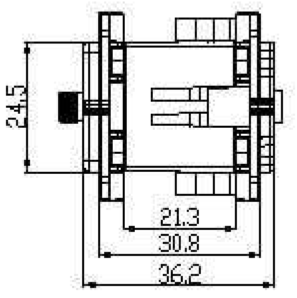
SR518 is mainly used for robots'joints, wheel,or caterpillar band, also can be used for simple position control.

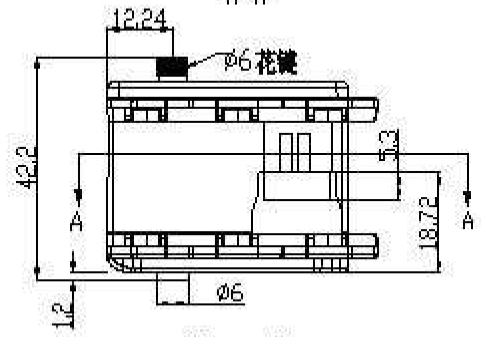
SR518 features:

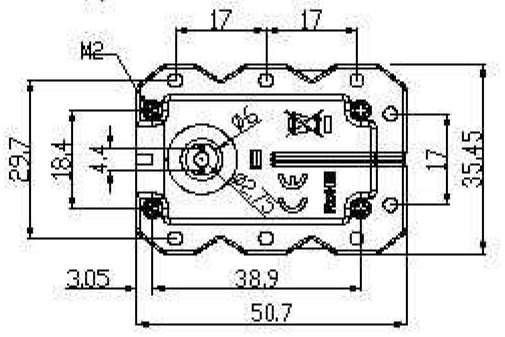
- ◆ High torque：18Kgf.cm
- ◆ Support wide voltage range DC 6.0V～12V
- ◆ Resolution: 0.32°
- ◆ Double sides installation,be suitable to install in robot joints
- ◆ High precision ,all metal gears, double bearing
- ◆ Good heal dissipation
- ◆ Rotation scope: 0-300°
- ◆ Can be 360deg.continuous rotation
- ◆ Bus connection,theoretically,254 units can be in series connection
- ◆ Communication baud rate : 1M
- ◆ 0.25Khz servo updated rate
- ◆ Be Compatible with Robotis Dynamixel communication protocols
- ◆ Feedback of position, temperature, speed and voltage etc.

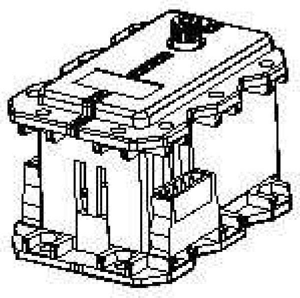
1.2 Diagram







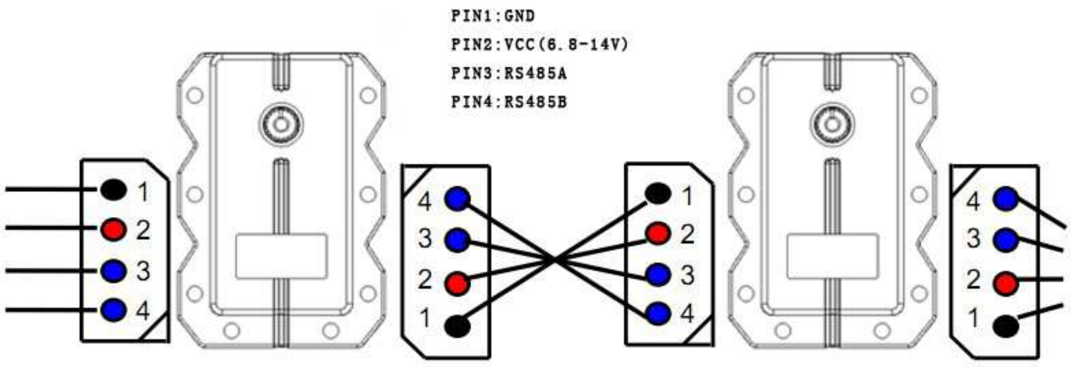




1.3 Electrical connection

1.3.1 Pin Definition

The following is the picture of SR518 electrical connection，Servos can be tandemed one by one via two groups of same pin definition terminals

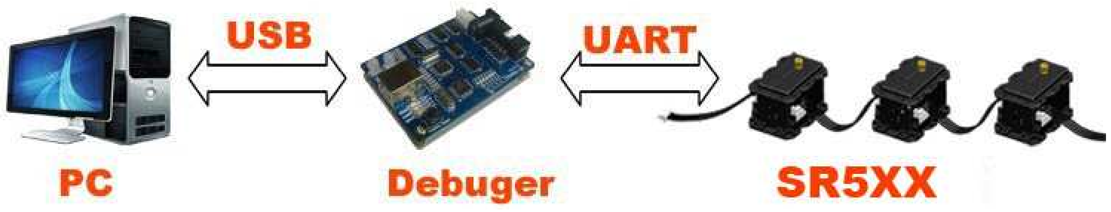


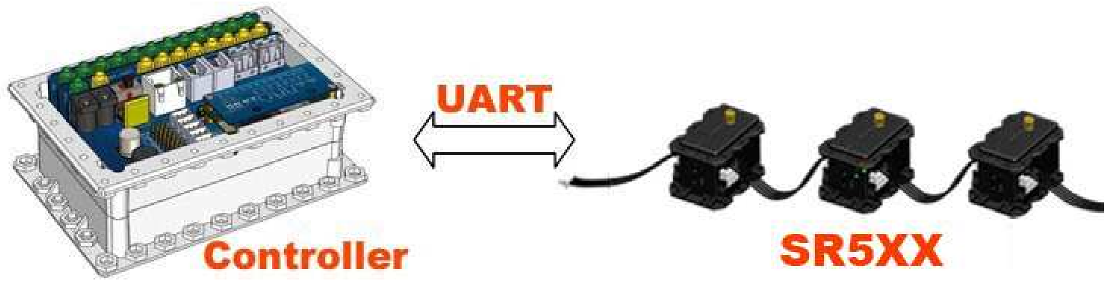
1.3.2 Servo Communication way

SR518 uses asynchronous serial bus communication mode, theoretically,at most 254 robot servos can be connected by Bus and to form a chain through UART asynchronous serial interface unified control. Every servo can set different node address, multiple servos can move together by unified singnal, also can move seperately via independent control.

SR518 communication instruction set to open, through asynchronous serial interface and the user's PC (controller or PC), you can set its parameters to change function control. Through asynchronous serial interface sends commands, SR518 can set for motor control mode or position control mode. In motor control mode, SR518 can be used as a DC decelerating motor with adjustable speed, In the position control mode, SR518 have 0-300 degrees of rotation, within the scope it has the exact position control performance with the adjustable speed.

Those half-duplex UART asynchronous serial interface ,as long as they accord with the agreement ,all can communicate with SR518 , and execute various control for SR518. Basically it has the following two types:





Model 1. through the debugger to control SR518

PC software through a serial send out a packet (accord with protocol format) , the debugger with serial interface transmitted to SR518. SR518 will execute the instruction of packet, and will return to respond the packet.

Robot Servo Terminal is a debugged software we ‘d like to recommend, user can also design an exclusive PC software according to the protocol of our manual

Model 2. through exclusive controller to control SR518

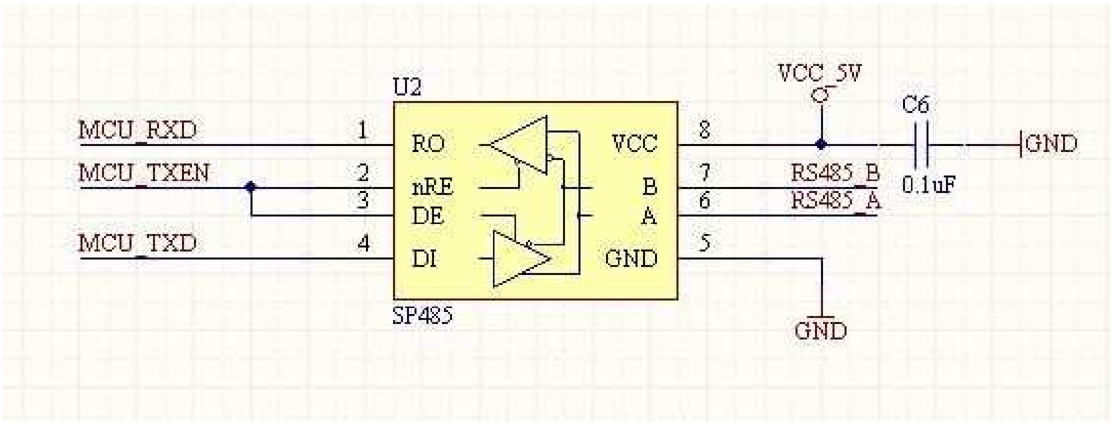
Model 1 can quickly debug SR5XX series robot servo, modify paramenters of function. However, this method can't work without PC, cannot build independent robots configurations. You can design a special controller, through the UART port of controller to control servo

1.3.3 subclause gives the UART interface schematic diagram of controller, chapter 5.gives the controller design specifications, according to this u can design ur own controller.

1.3.3 UART interface Schematic diagram

SR518 uses procedure code to control the time sequence of UART asynchronous serial interface, realizing half-duplex asynchronous serial bus communication, communication speed can reach to 1Mbps

In your own designed controller, UART interface used to communicate with SR518 must be installed as below



2. communication protocol

2.1 overview of communication

Controller and SR518 adopt Q&A (question and answer) way to communicate.

ID is a specific number for distinction of each SR518， when several SR518s are linked to one bus,eachSR518 matches an ID number,no SR518 with the same ID, main controller sends instruction packet which contains ID info to SR518,if the servo has ID number, then it can receive the instruction and return response/status packet

2.2 Instruction packet

Instruction packet format：

0XFF 0XFF	ID	Length	Instruction	Parameter1 ...Parameter N	Check Sum

0XFF 0XFF This signal notifies the beginning of the packet

ID It is the ID of SR518 which will receive Instruction Packet. It can use 254 IDs from 0 to 253 (0X00~0XFD). converts to a hexadecimal number. 0X00～0XFD。

Broadcasting ID : ID = 254 (0XFE)

If Broadcast ID is used, all linked SR518s execute command of Instruction Packet, and Status Packet is not returned.

LENGTH It is the length of the packet. The length is calculated as “the number of Parameters (N) + 2”.

PARAMETER0…N Parameter is used when Instruction requires ancillary data

CHECK SUM It is used to check if packet is damaged during communication. Check Sum is calculated according to the following formula.：

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)

When the calculation result of the parenthesis in the above formula is larger than 255 (0xFF), use only lower bytes.“~” Where, “~” is the Not Bit operator,but oppsite。

2.3 Status packet

SR518 executes command received from the Main controller and returns the result to the Main Controller. The returned data is called Status Packet. The structure of Status Packet is as follows:：

0XFF 0XFF	ID	Length	ERROR	Parameter1 ...Parameter N	Check Sum

ERROR It displays the error status occurred during the operatio of SR518. The meaning of each bit is described in the below table.：

Bit	Name	Contents
Bit 7	0	-
Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	Checksum Error	When the Checksum of the transmitted Instruction Packet is invalid

Bit 3	Range Erro	When the command is given beyond the range of usage
Bit 2	Overheating Erro	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Erro	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

2.4 Instruction type

This command gives an instruction to SR518 and has the following types.：

Instruction/name	Function	value	No.of parameters
PING	No execution. It is used when controller is ready to recevie Status Packet	0x01	0
READ DATA	This command reads data from SR518	0x02	2
WRITE DATA	This command writes data to SR518	0x03	2 or more
REG WRITE	it is similar to WRTE_DATA, but it remains in the standby state without being executed until the ACTION command arrives.	0x04	2 or more
ACTION	This command initiates motions registered with REG WRITE	0x05	0
RESET	his command restores the state of SR518 to the factory default setting.	0x06	0
SYNC WRITE	This command is used to control several SR518s simultaneously at a time	0x83	4 or more

2.4.1 WRITE DATA

Function	write data in SR518 control table
Length	N+3 (N is the length of write data)
Instruction	0X03
Parameter1	the first address of data write section
Parameter2	the first write data
Parameter3	the second data
ParameterN+1	the Nth data

eg1.(example) Set 1 to the ID of a random SR518

Keep the address of ID No. 3 in the control table,the write 1 in the address 3(the control table as below),the ID of instruction packet uses boradcast ID (0xFE)。

Instruction frame： 0XFF 0XFF 0XFE 0X04 0X03 0X03 0X01 0XF6

0XFF 0XFF	ID	Length	ERROR	Parameter1 ..Parameter N	Check Sum
0XFF 0XFF	0XFE	0X04	0X03	0X03 0X01	0XF6

Broadcast ID is used to send instructions,so no return status packet

Check Sum is calculated according to the following formula.：

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)

When the calculation result of the parenthesis in the above formula is larger than 255 (0xFF), use the lowest bytes.“~” Where, “~” is the Not Bit operator,but oppsite。

2.4.2 READ DATA

Function	Read data from the control table ofSR518
Length	0X04
Instruction	0X02
Parameter1	the first address of data read section
parameter2	the length of read data

Example2 Read the internal temperature of SR518(ID No. is 1)
。

Read one byte from address0X2B of the control table。

Instruction frame：0XFF 0XFF 0X01 0X04 0X02 0X2B 0X01 0XCC

T-head	ID	Valid data length	instruction	parameter	Check sum
0XFF 0XFF	0X01	0X04	0X02	0X2B 0X01	0XCC

Return data frame： 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDB

T-head	ID	Valid data length	Working state	parameter	Check sum
0XFF 0XFF	0X01	0X03	0X00	0X20	0XDB

If the read data is 0x20，the current temperature is about32℃（0x20）。

2.4.3 REG WRITE

REG WRITEinstruction is similar to WRITE DATA，just the execution time different。When receive the REG WRITE instruction frame，store the received data in buffer to spare/backup，and set 1 to REG register/Registered Instruction（Address 0x2c）。when receive the ACTION instruction，the stored instruction will be executed eventually。

Length	N+3 (N is the number of write data)
Instruction	0X04
Parameter1	the first address of data write section
Parameter2	the first data to write
Parameter3	the second data to write
ParameterN+1	the Nth data to write

2.4.4 ACTION

Function	trigger REG WRITE instruction
Length	0X02
Instruction	0X05
Parameter	-

ACTION instruction is very important/useful in the control of many SR518s at the same time when control many independent SR518s, use ACTION instruction can make the first and last SR518 simultaneously execute their own actions, no delay.Send ACTION instructions to mutiple SR518s，broadcastID（0xFE）must be used,therefore，no return data frame。

2.4.5 PING

Function read the working state of SR518
Length 0X02
Instruction 0X01
Parameter -
Example 3 Read the working state of SR518(ID No. is 1)
。
Instruction frame：0XFF 0XFF 0X01 0X02 0X01 0XFB`

T-head	ID	Valid data length	instruction	Check sum
0XFF 0XFF	0X01	0X02	0X01	0XFB

Return data frame： 0XFF 0XFF 0X01 0X02 0X00 0XFC

T-head	ID	Valid data length	Working state	Check sum
0XFF 0XFF	0X01	0X02	0X00	0XFC

No matter broadcastID or Return Level (Address 16),when they equal to 0，as long as the PINE instruction and Check Sum are correct，the working state of SR518 will return。

2.4.6 RESET

Function Reset the data in control table to factory value
Length 0X02
Instruction 0X06
Parameter -
Example 4 Reset SR518，ID No.is 0。
Instruction frame：0XFF 0XFF 0X00 0X02 0X06 0XF7`

T-head	ID	Valid data length	Working state	parameter
0XFF 0XFF	0X00	0X02	0X06	0XF7

返回的数据帧：0XFF 0XFF 0X00 0X02 0X00 0XFD

T-head	ID	Valid data length	Working state	parameter
0XFF 0XFF	0X00	0X02	0X00	0XFD

2.4.7 SYNC WRITE

Function	Control mutiple SR518s at the same time。
ID	0XFE
Length	(L + 1) * N + 4 (L: the data length of every SR518 N: how many SR518)
指令	0X83
Parameter1	the first address of data write section
parameter2	the length of write data(L)
parameter3	the ID No. of first SR518
parameter4	the first write data of first SR518
parameter5	the second write data of the first SR518
...	
parameter L+3	**the Lth write data of the first SR518**
parameter L+4	the ID No. of second SR518
parameterL+5	**the first write data of the second SR518**

parameter L+6　the second write data of the second SR518

…

parameter2L+4　the Lth write data of the second SR518

….

Different from the REG WRITE + Action instruction is that it's much faster.one SYNC WRITE instruction can modify the control table content of mutiple SR518s at one time,but REG + Action must take several steps to finish this instruction. Nonetheless, when use SYNC WRITE instructions,the write data length and the first address of stored data　must be the same ,that mutiple SR518s must execute the same actions

Example5 write position and speed for 4 SR518s below

ID0: position：0X010；speed:0X150

ID1: position：0X220；speed:0X360

ID2: position：0X030；speed:0X170

ID3: position：0X220；speed:0X380

Instruction frame : 0XFF 0XFF 0XFE 0X18 0X83 0X1E 0X04 0X00 0X10 0X00 0X50 0X01 0X01 0X20 0X02 0X60 0X03 0X02 0X30 0X00 0X70 0X01 0X03 0X20 0X02 0X80 0X03 0X12

T-head	ID	Valid data length	instruction	parameter	Check sum
0XFF	0XFE	0X18	0X83	0X1E 0X04	0X12
0XFF				0X00 0X10 0X00 0X50	
				0X01 0X01 0X20 0X02	
				0X60 0X03 0X02 0X30	
				0X00 0X70 0X01 0X03	
				0X20 0X02 0X80 0X03	

Broadcast ID is used,so no return status packet。

Chapter 3 Control table

3.1 control table

The robot servo itself information and control parameters formed a table, stored in its control chip RAM and EEPROM area. Through the content revise at any time so that we can control the servo at any time.This table is called Memory Control table,the content as below：

Address hexadecima	Description	Read write	Initial Value (Hexadecimal)	Store area
0（0x00）	--	--	--	EEPROM
1（0x01）	--	--	--	
2（0x02）	Version of Firmware	read	--	
3（0x03）	ID	read/write	1（0x01）	
4（0x04）	Baud Rate	read/write	1（0x01）	
5（0x05）	Return Delay Time	read/write	0（0x00）	
6（0x06）	Lowest byte of clockwise Angle Limit（L）	read/write	0（0x00）	
7（0x07）	Highest byte of clockwise Angle Limit（H）	read/write	0（0x00）	
8（0x08）	Lowest byte of counterclockwise Angle Limit（L）	read/write	255（0xFF）	
9（0x09）	Hightest byte of counterclockwise Angle Limit（H）	read/write	3（0x03）	
10（0x0A）	--			
11（0x0B）	the Highest Limit Temperature	read/write	80（0x50）	
12（0x0C）	Lowest Limit Voltage	read/write	?	
13（0x0D）	Highest Limit Voltage	read/write	?	
14（0x0E）	Max Torque(L)	read/write	255（0xFF）	
15（0x0F）	Max Torque(H)	read/write	3（0x03）	
16（0x10）	Status Return Leve	read/write	2（0x02）	
17（0x11）	Alarm LED	read/write	37（0x25）	
18（0x12）	Unload condition	read/write	4（0x04）	
19（0x13）	--	--	--	
20（0x14）	Potentiometer correction	--	--	
21（0x15）	Potentiometer correction	--	--	
22（0x16）	Potentiometer correction	--	--	
23（0x17）	Potentiometer correction	--	--	
24（0x18）	Torque On/Off	read/write	0（0x00）	RAM
25（0x19）	LED On/Off(switch)	read/write	0（0x00）	
26（0x1A）	CW Compliance margin	read/write	2（0x02）	
27（0x1B）	CCW Compliance margin	read/write	2（0x02）	

28（0x1C）	Clock wise ratio	read/write	32（0x20）	
29（0x1D）	Counter clock wise ratio	read/write	32（0x20）	
30（0x1E）	Goal Position(（L）	read/write	[Addr36]value	
31（0x1F）	Goal Position(（H）	read/write	[Addr37]value	
32（0x20）	Moving Speed（L）	read/write	0	
33（0x21）	Moving Speed（H）	read/write	0	
34（0x22）	Accelerating speed	read/write	32	
35（0x23）	Decelerating speed	read/write	32	
36（0x24）	Present Position(（L）	read	?	
37（0x25）	Present Position(（H）	read	?	
38（0x26）	Present Speed（L）	read	?	
39（0x27）	Present Speed（H）	read	?	
40（0x28）	Present Load	read	?	
41（0x29）	Present Load	read	?	
42（0x2A）	Present Voltage	read	?	
43（0x2B）	Present Temperature	read	?	
44（0x2C）	REG WRITE mark	read	0（0x00）	
45（0x2D）	--		0（0x00）	
46（0x2E）	In operation	read	0（0x00）	
47（0x2F）	Lock mark	read/write	0（0x00）	
48（0x30）	minimumPWM(L)	read/write	90（0x5A）	
49（0x31）	minmumPWM(H)	read/write	00（0x00）	

If the control parameters have "L" and "H" 2 different commands, the scope is 0x00 to 0x3ff; the command of Only one byte parameters, the scope is 0x00 to 0xff.

Parameters stored in the RAM area will be not saved when the electricity/power is shut off, but parameters stored in EEPROM area can be saved. '--' stands for unchangable parameters

Detail description as follows：

0x03：

Keep the ID No. of SR518。

0x04:

Keep BPS(baud rates) caculation parameters

Baud rate formula： Speed(BPS) = 2000000/(Address4+1)。

When the Address4 is default3，the BPS is 500K，According to the formula ， users can modify baud rates to other ones they need, but after the power, only
1M ,250K ,500K ,115200,57600,19200 will be saved. Other baud rates will be restored to 1M

Address4 in the formula is the saved data of address 0x04。The comparision of Baud rate and the corresponding calculation parameters as:

Address4	Hex	practicalBPS	GoalBPS	Error tolerance
3	0x03	500000.0	500000.0	0.000%
7	0x07	250000.0	250000.0	0.000%
16	0x10	117647.1	115200.0	-2.124%
34	0x22	57142.9	57600.0	0.794%
103	0x67	19230.8	19200.0	-0.160%

0x05:

Set return delay time，When SR518 receive an instruction which needs response, u can set the time of how long can be delayed to respond . Time range: parameters（0~255）*2US,if the parameter is 250, it will respond after500us ;when the parameter is 0, signify that SR518 will response in the shortest time , 8us is the minimum reaction time, so the actual minimum response time is 8us

0x06～0x09:

Set the Angle scope of servoSR518can run,clockwise angle limit=< goal angle value≤<= counter colockwise limit value

Attention !clockwise angle limit value must less than counter clockwise angle limit value.If the goal angel value supass the range means that it equals to limit value，and the Angle overscope symbol of servo SR518 status will be 1 。

0x0B

The highest working temperature is 80 degree,users can’t change

0x0E～0x0F:

Set the maximum output servo torque . 0X03FF corresponding SR518 the maximum output capacity.

0x10:

The level of response,set whether the servo SR518 will return the data or not after receiving datum

Address16	Return status packet

0	No return to all instructions
1	Only return to read instructions
2	Return to all instructions

0x11:
Set LED twinkle alarm conditions

BIT	Function
BIT7	0
BIT6	If the bit is set as 1，the LED flickers when instruction error happened
BIT5	If the bit is set as 1，the LED flickers when overload error happened
BIT4	If the bit is set as 1，the LED flickers when checksum error happened
BIT3	If the bit is set as 1，the LED flickers when the command is given beyond the range of usage
BIT2	If the bit is set as 1，the LED flickers when Overheating error happened
BIT1	If the bit is set as 1，the LED flickers when Angle Limit Error happened
BIT0	If the bit is set as 1，the LED flickers when Input Voltage Error happened

If all that happens at the same time,please follow the logic "or"principle

0x12:
Setting unload conditions

BIT	Function
BIT7	0
BIT6	If the bit is set as 1，unload when instruction error happened
BIT5	--
BIT4	If the bit is set as 1，unload when checksum error happened
BIT3	If the bit is set as 1，unload when the command is given beyond the range of usage
BIT2	If the bit is set as 1，unload when Overheating error happened
BIT1	If the bit is set as 1，unload when Angle Limit Error happened
BIT0	If the bit is set as 1，unload when Input Voltage Error happened

If all that happens at the same time,please follow the logic "or"principle

BIT5 overload mark is invalid, when SR518 overload, torque force will automatically decline to a safety value without burning down/out, not completely unload

0x18:
Torque output switch，"1"on，"0"off。
0x19:
LED torque，"1"开 on，"0"off。
0x1A～0x1B:
The size of position closed loop dead zone
0x1C～0x1D:

The P parameter of position closed loop effect the adjusting speed of position loop.Generally, the user does not need to modify the parameters, ACC,DCC controls the smoothless of movements. Namely, acceleration and deceleration,set By 0x22 ~ 0x23

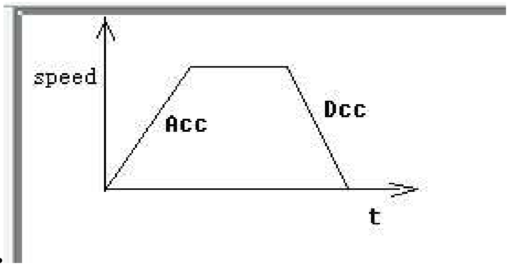
0x1E～0x1F:

Position the servo runs to when the command is given，range0x0000—0x03ff,0x0000 correspond to 0 degree，0x03ff correspond300 degree, deviation±2%。

0x20～0x21:

Set the speed of target location servo runs to. When the location is set as1023,the corresponding maximum speed of SR518 is62RPM.

under Position model,the minimum speed is 1, the maximum speed is 0 ,equivalent to 1023.



0x22～0x23:

The running accelerating and decelerating speed of servo (ACC,Dcc).range 0~255。

0x24～0x2E:

The running state of servo, such as current position, speed, etc, just can read can't write.

0x2C：

If there r REG WRITE instructions waiting for execution, then it displays 1, when the REG WRITE instructions have been executed completely,it displays 0。

0x2E：

It displays 1 if the servoSR518 is in the operation , otherwise it displays 0.

0x2F：

Lock function location. If this bit is set as 1, only 0x18 ~ 0x23 can be modified/changed and other locations are locked cannot be modified. Once the lock function go into effect, the electricity/power must be shut down then it will lose effectieness

0x30～0x31:

PWM ZhanKongBi(duty cycle)'s minimum output value.

3.2 Motor speed adjustment mode

The SR5XX series robot servos can switch over to motor speed adjustment mode, can be used on the actuators of wheel and crawler turnover .The clockwise and anti-clockwise Angle restrictions (0x06 ~ 0x09) r both set to 0, give a another speed（0x20～0x21), the servo will run turn in motor speed adjustment mode. Speed has magnitude and direction control mode, are shown below

:

BIT	11~15	10	9	8	7	6	5	4	3	2	1	0
VALUE	0	0/1	SPEED VALUE									

address**0x20～0x21：**BIT10 is direction bit，when it equals to 0,the servo rotates anti-clockwise，when is 1,servo rotates clockwise。BIT0～BIT9 r size bits。

Attention：under Motor model,acceleration equals to deceleration,and set by 0x22

4. Examples

- **Example 1:Change the servo’s ID number 1 into 0**

Instructions = WRITE DATA； Address = 0X03; Data = 0x00
Instruction packet： FF FF 01 04 03 03 00 F4
Status packet： FF FF 01 02 00 FC
state： no error

Attention: except set the same ID address for multiple servos at the same time, servos need to be set ID No. one by one. The instruction packet is the control data that main controller or debugger sends to SR518 ,the status packet of the information SR518 refers to return.

example2：restrict the running angle scope of the(ID No. is 0)servo in 0 ~ 150°
Instruction = WRITE DATA； Address = 0X08; Data = 0XFF, 0X01
Instruction packet： FF FF 00 05 03 08 FF 01 EF
Status packet： FF FF 00 02 00 FD
State： no error

- **Example3： Set the working temperature highest limit of No. 0 servo to 80degree**

Instruction = WRITE DATA； Address = 0X0B; Data = 0X50
Instruction packet： FF FF 00 04 03 0B 50 9D
Status packet： FF FF 00 02 00 FD
State： no error

- **Example 4： Set the recommended working voltage range of No.0 servo as 6v-9v**

6Vuse60（0X3C）to espress，9Vuse90（0X5A）to express
Instruction = WRITE DATA； Address = 0X0C; Data = 0X3C, 0X5A
Instruction packet： FF FF 00 05 03 0C 3C 5A 55
Status packet： FF FF 00 02 00 FD
State： no error

- **Example5： Set the output torque limit of No. 0 servo as the half maximum**

When output the largest,correspond to 0x03ff， so output half,correspond to 0x01ff
Instruction = WRITE DATA； Address = 0X0E; Data = 0XFF, 0X01
Instruction packet： FF FF 00 05 03 0E FF 01 E9
Status packet： FF FF 00 02 00 FD
State： no error

- **Example 6： Set No.0 servodo not return data for all instructions**

Instruction = WRITE DATA； Address = 0X10; Data = 0X00
Instruction packet： FF FF 00 04 03 10 00 E8
Status packet： FF FF 00 02 00 FD
State： no error

- **Example7： Unload the No. 0 servo**

Instruction = WRITE DATA； Address = 0X18; Data = 0X00
Instruction packet： FF FF 00 04 03 18 00 E0
Status packet： FF FF 00 02 00 FD

State：no error

- **Exampple8：Let the No.0 servo run to the position of 150°in medium speed**

Full speed,correspond to 0x03ff,so mid-speed can correspond to 0x01ff，300°to 0x03ff，150°to 0x01ff

Instruction = WRITE DATA；Address = 0X1E; Data = 0x00, 0x02, 0x00, 0x02
Instruction packet：FF FF 00 07 03 1E 00 02 00 02 D3
Status packet：FF FF 00 02 00 FD
State：no error

- **Example9：Let the No.2 servo run to the position of 0°**

Attention:Require them to run at the same time

Use REG_WRITE + ACTION instruction can let them run at the same time
ID=2；Instruction = REG_WRITE；Address = 0X1E; Data = 0x00, 0x00
ID=1；Instruction = REG_WRITE；Address = 0X1E; Data = 0xFF, 0x03
ID=0XFE; Instruction = ACTION

Instruction packet：FF FF 02 05 04 1E 00 00 D6
Status packet：FF FF 02 02 00 FB
Instruction packet：FF FF 01 05 04 1E FF 03 D5
Status packet：FF FF 01 02 00 FC
Instruction packet：FF FF FE 02 05 F
Broadcast ID is used,no return status packet
State：no error

- **Example10: Set only the address section 0x18～0x23 can be modified of No.0 servo in control table,that is write 1 in the address 0x18～0x23**

Instruction = WRITE DATA；Address = 0X2F; Data = 0x03
Instruction packet：FF FF 00 04 03 2F 01 C8
Status packet：FF FF 00 02 00 FD
State：no error

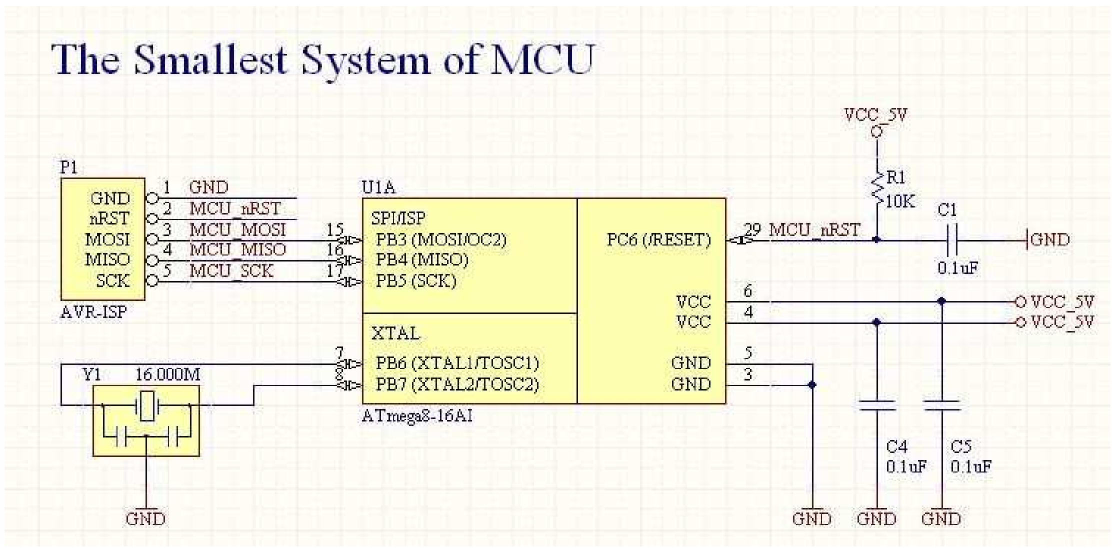
- **Example 11：　When the No.0 servo　is in position closed loop operation, set the ACC as 4,Dcc as 6,the maximum value of ACC and Dcc as 255**

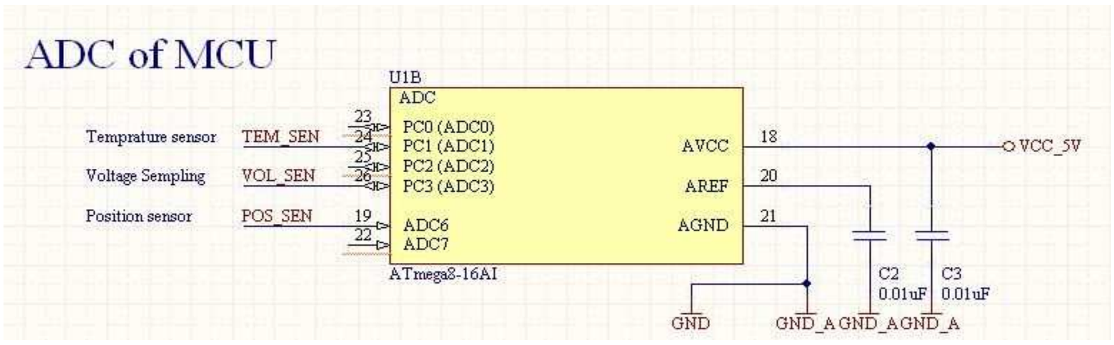
Instruction = WRITE DATA；Address = 0X30；Data = 0X04, 0X06
Instruction packet：FF FF 00 05 03 30 04 06 BD
Status packet：FF FF 00 02 00 FD
State：no error

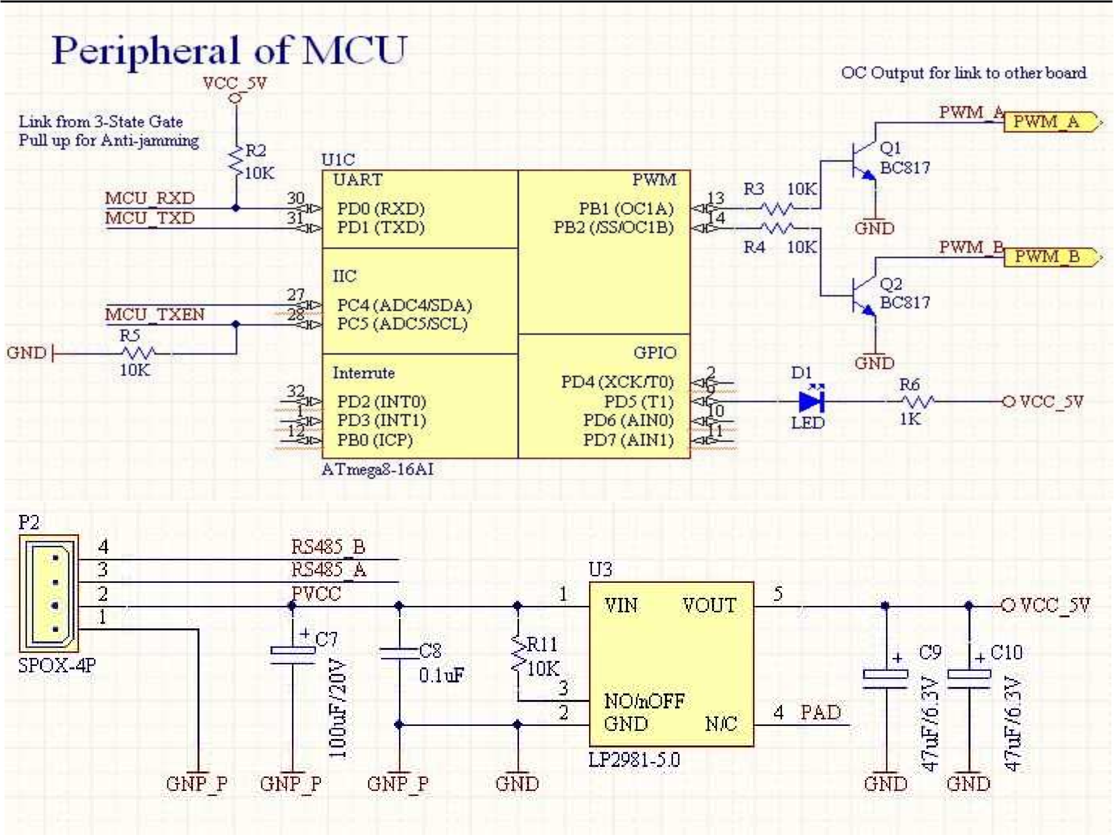
Chapter 5 The development of servo controller based on AVR SCM (Single Chip Micyoco)

5.1 Schematic diagram

In actual use of SR518 , you may need to develop your own controller to control the servo. Below is a brief schematic diagram of servo control card, you can refer to the diagram to design your own control.





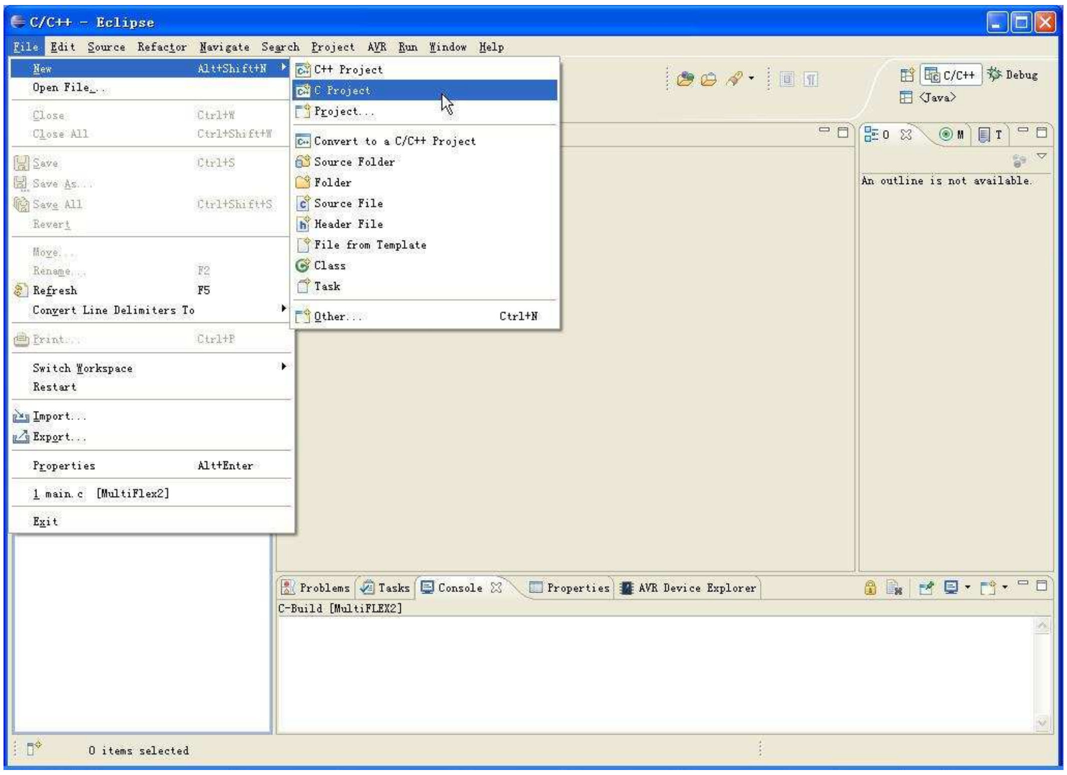


5.2 Control card program development

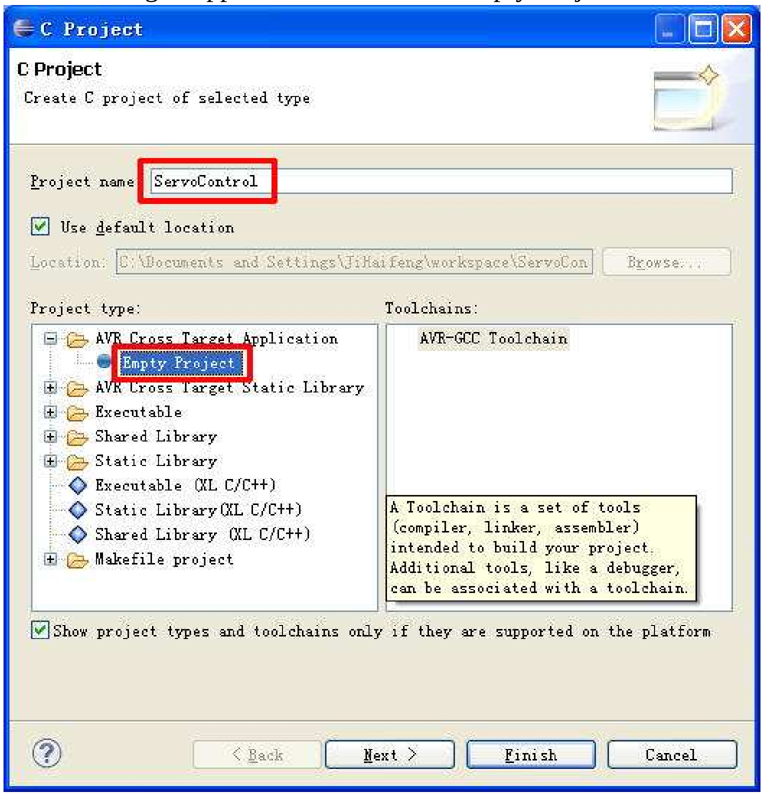
Here we use Eclipse as development environment, if you are not familiar with the Eclipse development program AVR, please refer to the document "EclipseForAVR program development." The initial goal of this program is to make the servo sway between two locations, then by adding code, can realize the control logic of obstacle-avoidance car through two infrared sensors obstacle-avoidance car. Following the development process.

5.21 Create engineering

Run Eclipse，Choose“File”->”New”->”C Project” Menu item，new project，as following。



In the popup dialog box type “Servo Control” in “Project Name” ， in“Project Type”listing, the item of “AVR Cross Target Application”，Choose“Empty Project”，click“Next”。


C Project

Create C project of selected type

Project name: ServoControl

[✓] Use default location

Location: C:\Documents and Settings\JiHaifeng\workspace\ServoCon

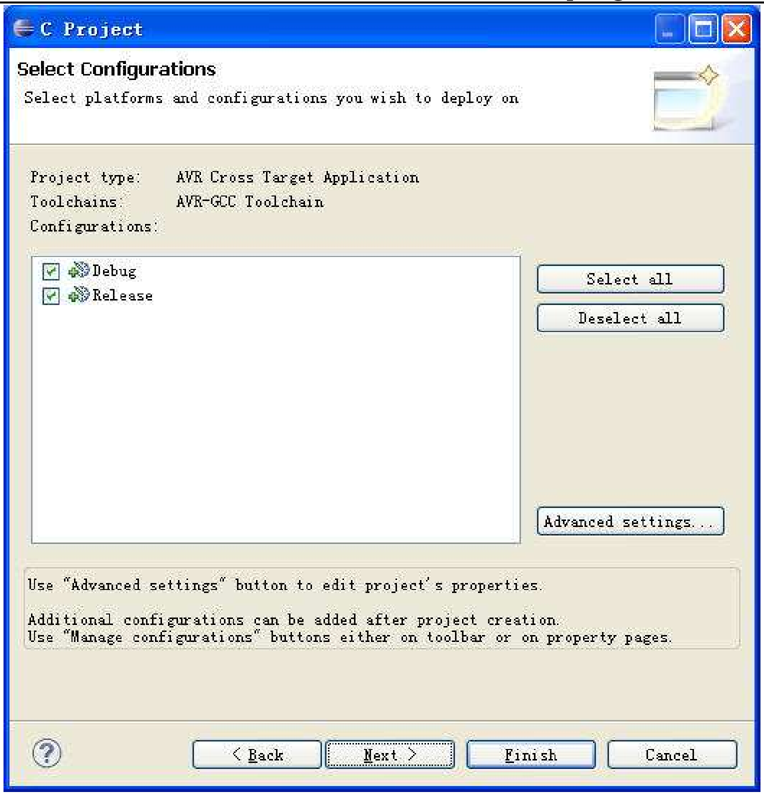
Project type:
- AVR Cross Target Application
  - Empty Project
- AVR Cross Target Static Library
- Executable
- Shared Library
- Static Library
- Executable (XL C/C++)
- Static Library (XL C/C++)
- Shared Library (XL C/C++)
- Makefile project

Toolchains:
AVR-GCC Toolchain

A Toolchain is a set of tools (compiler, linker, assembler) intended to build your project. Additional tools, like a debugger, can be associated with a toolchain.

[✓] Show project types and toolchains only if they are supported on the platform

Pop-up the option configuration dialog as below，click“Next”。



C Project

Select Configurations

Select platforms and configurations you wish to deploy on

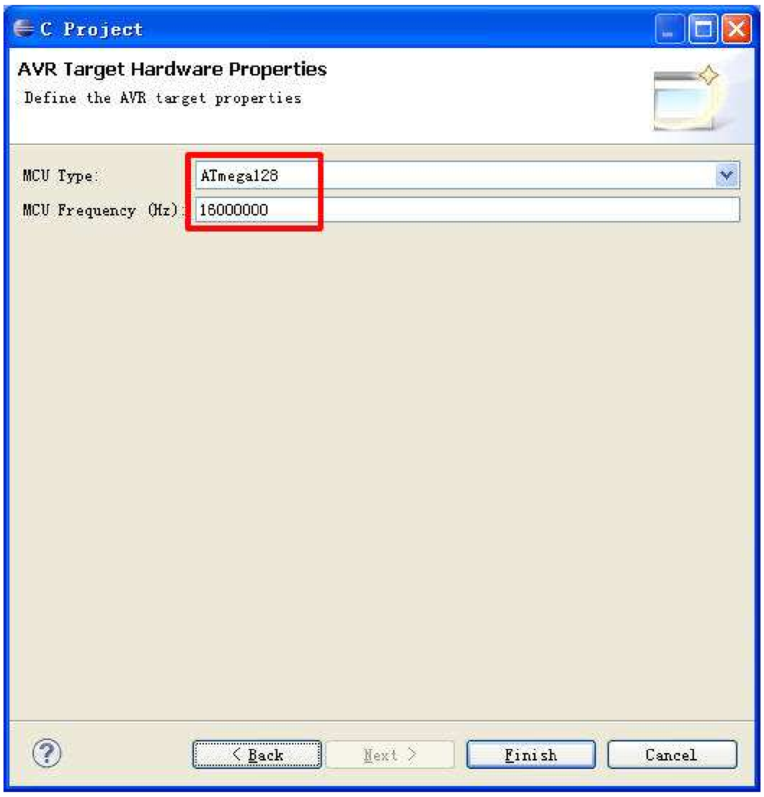
Project type: AVR Cross Target Application
Toolchains: AVR-GCC Toolchain
Configurations:

- [✓] Debug
- [✓] Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

Pop-up target equipment attribute Settings dialog，in“MCU Type”listing,pls choose“ATmega128”，and type “16000000”in“MCU Frequency” as below。click“Finish”



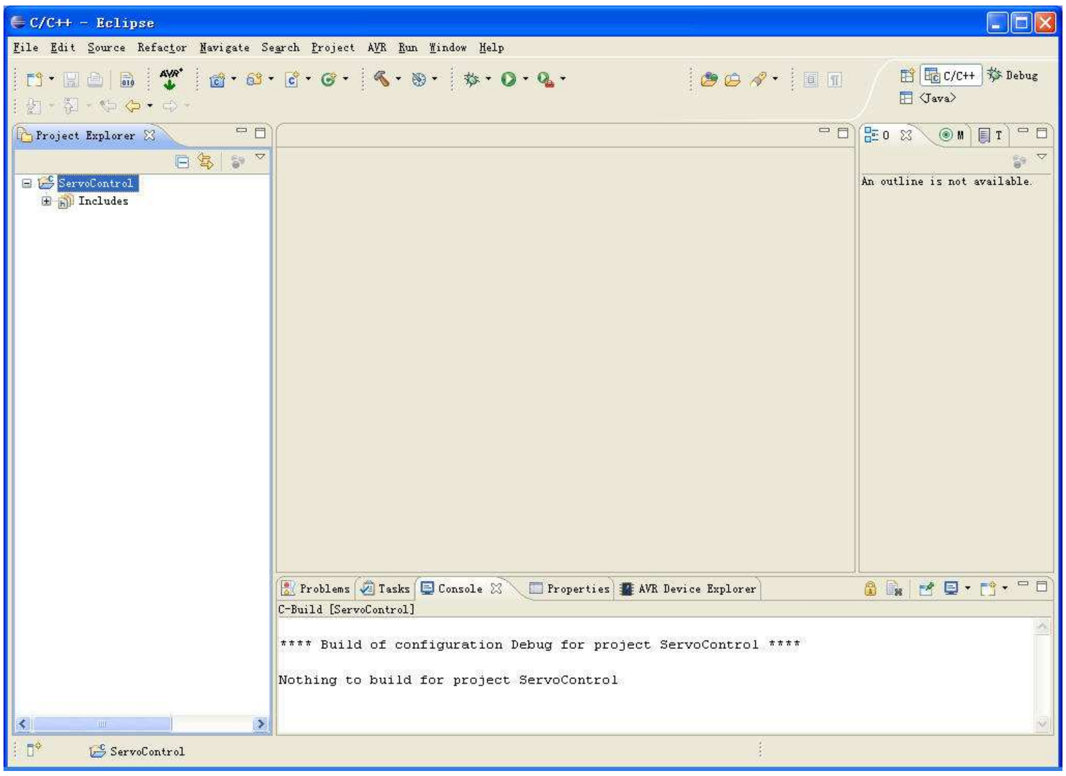
C Project

AVR Target Hardware Properties

Define the AVR target properties

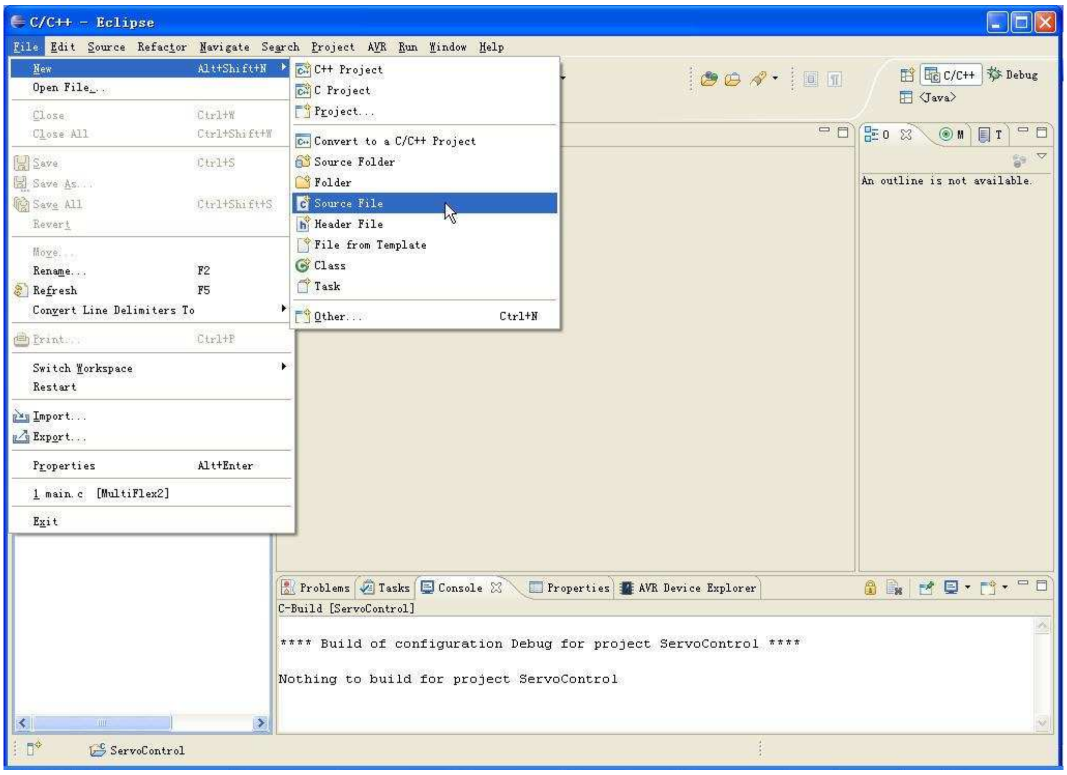
MCU Type: ATmega128
MCU Frequency (Hz): 16000000

Return to Eclipse main window，in the left“Project Explorer”item，display“ServoControl”program as below：



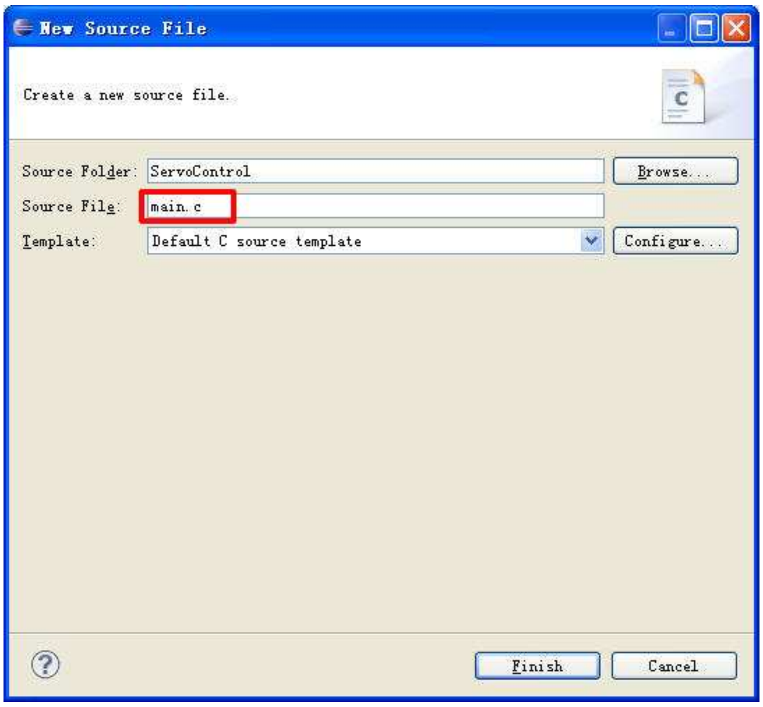
```
C-Build [ServoControl]

**** Build of configuration Debug for project ServoControl ****

Nothing to build for project ServoControl
```

Choose“File”->”New ”->”Source File”menu item，Construct new source code files as below：



```
C-Build [ServoControl]

**** Build of configuration Debug for project ServoControl ****

Nothing to build for project ServoControl
```

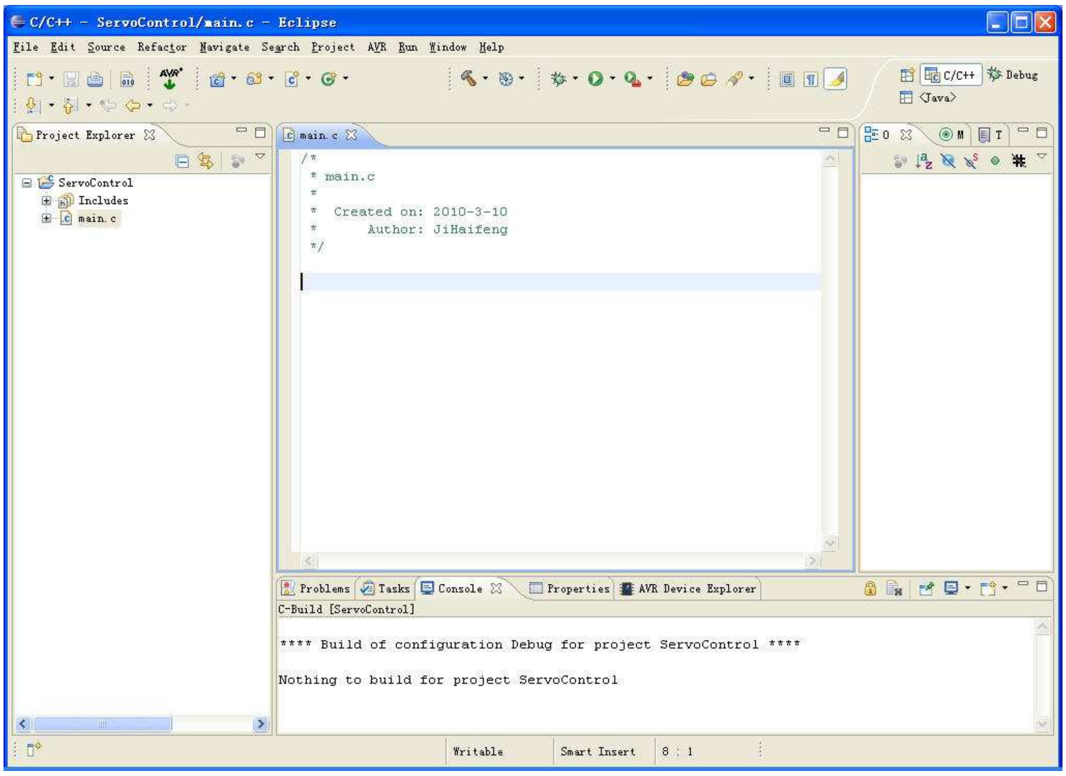
Pop-up setting dialogbox，type “main.c”in “Source File”as below,click“Finish”



Create a new source file.

Source Folder: ServoControl
Source File: main.c
Template: Default C source template

Return to Eclipsemain window , main.c,documents to be automatically added to“ServoControl”and in editable form：


C/C++ - ServoControl/main.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
ServoControl
Includes
main.c
C/C++
Debug
<Java>
Problems Tasks Console Properties AVR Device Explorer
Writable Smart Insert 8 : 1

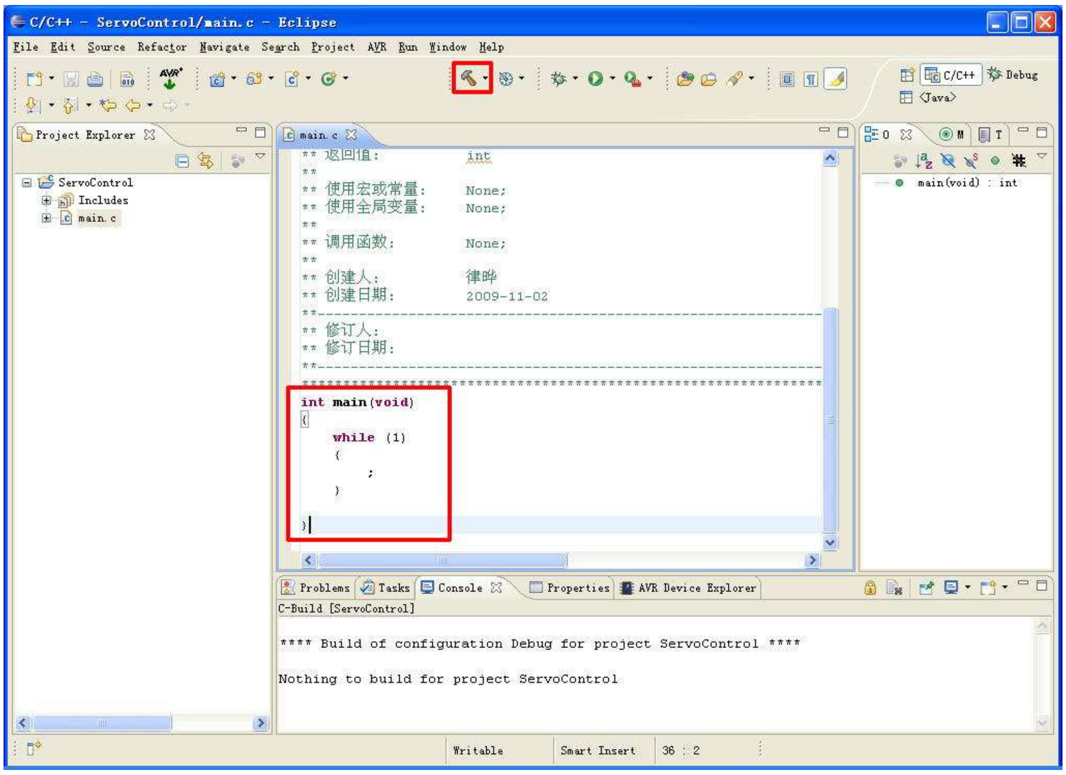
```
/*
 * main.c
 *
 *  Created on: 2010-3-10
 *      Author: JiHaifeng
 */
```

```
C-Build [ServoControl]

**** Build of configuration Debug for project ServoControl ****

Nothing to build for project ServoControl
```

Add main function in main.c window as below。The click start compiler.main function code is below：

/********************************Copyright*************************************
** Beijing UPTECH Robotics Co.Ltd.
** yanfabu@126.com

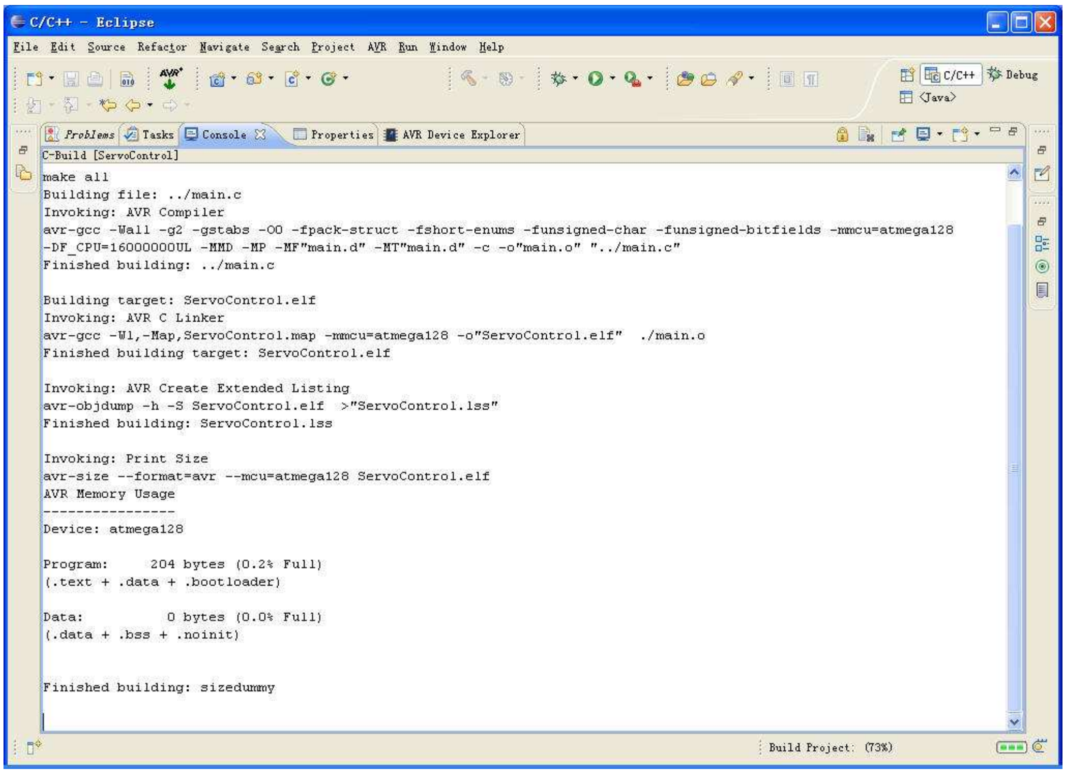
```
**                          http://robot.up-tech.com
**
**-----------------------------------------------File
Info-----------------------------------------------------------
** (File Name):                          main.c
** (Date Last Modified):            2010-3-25
** (Last Version):                       1.0
** (Description):
**
**---------------------------------------------------------------------------------------------------------------
----------
** (Author):                             robot
** (Date Created):                       2010-3-25
** (Version):                                1.0
** (Description):
**
*****************************************************************************/
int main(void)
{
        while (1)
        {
                ;
        }
}
```



```
** 返回值:          int
**
** 使用宏或常量:    None;
** 使用全局变量:    None;
**
** 调用函数:        None;
**
** 创建人:          律晔
** 创建日期:        2009-11-02
**------------------------------------------------------------------
** 修订人:
** 修订日期:
**------------------------------------------------------------------
********************************************************************
int main(void)
{
    while (1)
    {
        ;
    }

}
```

```
C-Build [ServoControl]

**** Build of configuration Debug for project ServoControl ****

Nothing to build for project ServoControl
```

Compile information appear in“Console”child window，double click“Console”tag maxmium window to check aa follows。U can see compile correct or not。


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
C/C++ Debug <Java>
Problems Tasks Console Properties AVR Device Explorer
Build Project: (73%)

```
C-Build [ServoControl]
make all
Building file: ../main.c
Invoking: AVR Compiler
avr-gcc -Wall -g2 -gstabs -O0 -fpack-struct -fshort-enums -funsigned-char -funsigned-bitfields -mmcu=atmega128
-DF_CPU=16000000UL -MMD -MP -MF"main.d" -MT"main.d" -c -o"main.o" "../main.c"
Finished building: ../main.c

Building target: ServoControl.elf
Invoking: AVR C Linker
avr-gcc -Wl,-Map,ServoControl.map -mmcu=atmega128 -o"ServoControl.elf"  ./main.o
Finished building target: ServoControl.elf

Invoking: AVR Create Extended Listing
avr-objdump -h -S ServoControl.elf  >"ServoControl.lss"
Finished building: ServoControl.lss

Invoking: Print Size
avr-size --format=avr --mcu=atmega128 ServoControl.elf
AVR Memory Usage
----------------
Device: atmega128

Program:     204 bytes (0.2% Full)
(.text + .data + .bootloader)

Data:          0 bytes (0.0% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

Double click“Console”tag again，back to the state before 。

5.22 Add initialization function

Control card through UART0 communication to control servos，after the program run,first we must initialize the UARTO。Besides,when realizing obstacle-avoiding car control logic，need to connect two IO quantity sensors to judge obstacles, so also needs to initialize IO。

Add InitGpiofunction here to realize the initialization of IO，the code is below：

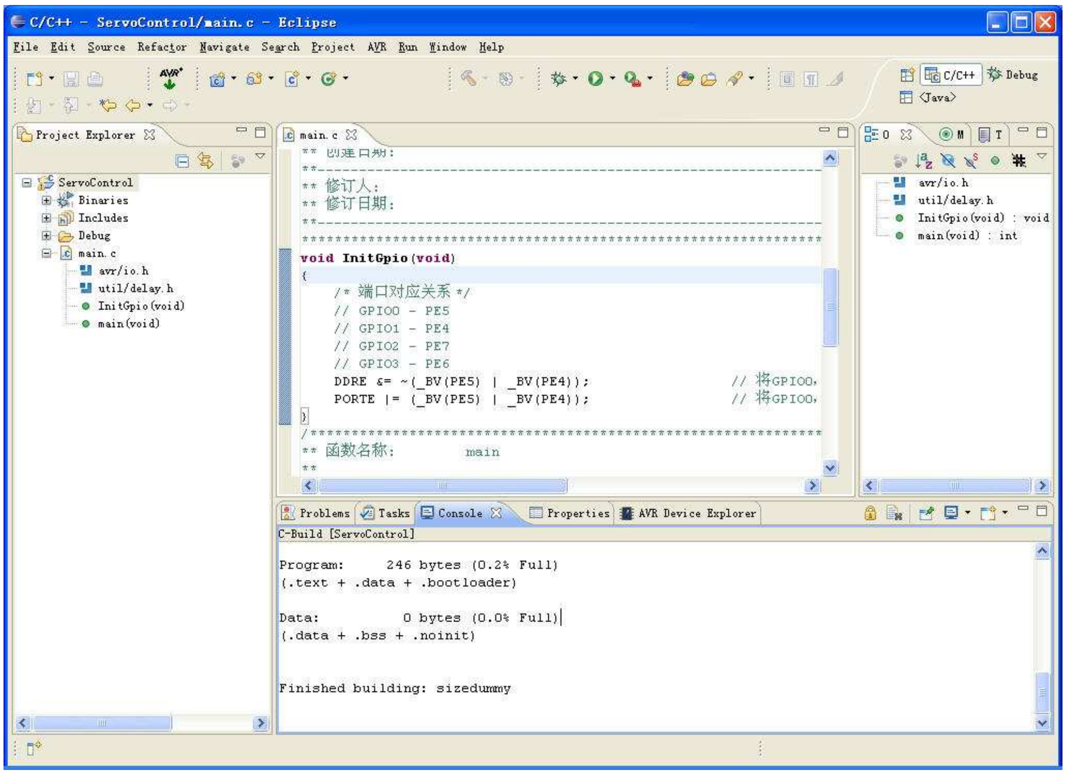
```
/******************************** Function Info *************************
** (Function Name):                  InitGpio
**
** (Description):                                        (Init general digital I/O port)
**
**
** (Input Variable):                 void
** (Return Value):                       void
**
** (Macro or constant used):  None
** (Global variable used):    None
**
** (Other function called):          None
**
** (Author):                             robot
** (Date created):                   2010-3-25
**
```

```
** (Revised):
** (Date Revised):
**************************************************************************/
void InitGpio(void)
{
        /*（Port map）  */
        // GPIO0 - PE5
        // GPIO1 - PE4
        // GPIO2 - PE7
        // GPIO3 - PE6

        // (Set general digital I/O port 0 and port 1 to input mode)
        DDRE &= ~(_BV(PE5) | _BV(PE4));



        // (Set Set general digital I/O port 0 and pull-up state)
        PORTE |= (_BV(PE5) | _BV(PE4));
}
```

Duplicate & paste the above code to the main.c documents,before main function，add the need header files in the main.c documents starting position：

```
#include <avr/io.h>
#include <util/delay.h>
```

Click keep main.c，then click recompile procedure，u can see the compilation result in the “Console“child window as below。



```
** 创建日期:
**-----------------------------------------------------------------
** 修订人:
** 修订日期:
**-----------------------------------------------------------------
*****************************************************************
void InitGpio(void)
{
    /* 端口对应关系 */
    // GPIO0 - PE5
    // GPIO1 - PE4
    // GPIO2 - PE7
    // GPIO3 - PE6
    DDRE &= ~(_BV(PE5) | _BV(PE4));                    // 将GPIO0,
    PORTE |= (_BV(PE5) | _BV(PE4));                    // 将GPIO0,
}
/****************************************************************
** 函数名称:        main
**
```

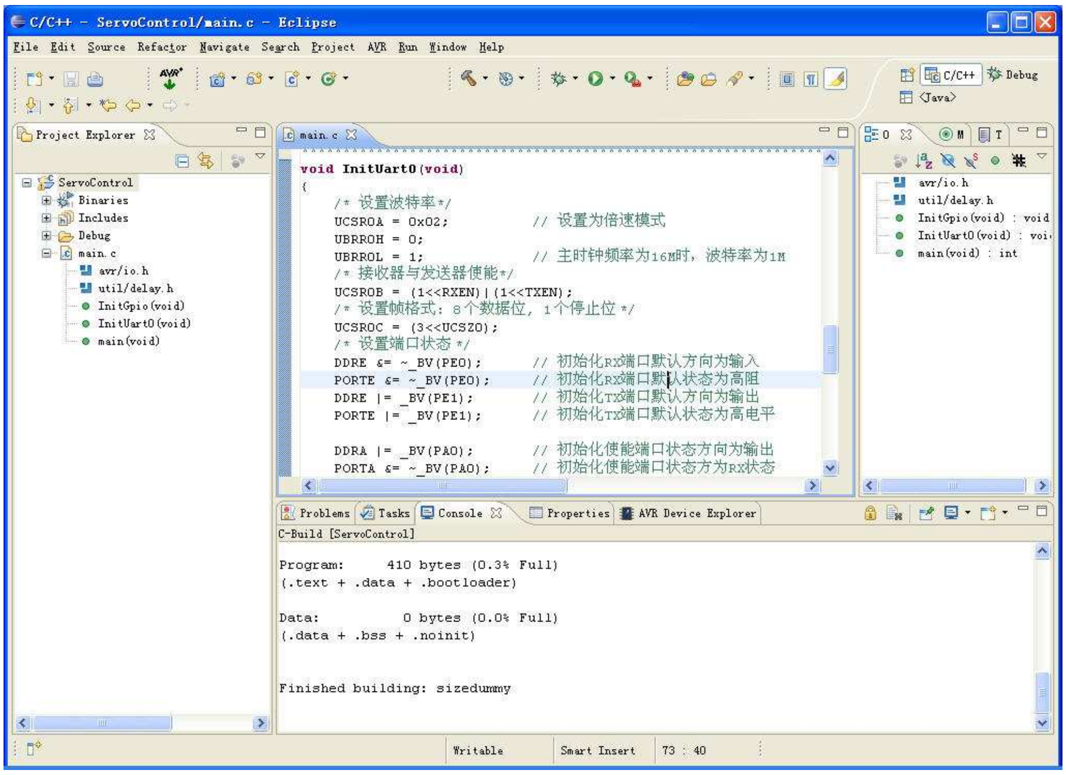
```
C-Build [ServoControl]

Program:     246 bytes (0.2% Full)
(.text + .data + .bootloader)

Data:          0 bytes (0.0% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

Add function InitUart0 in main.c to realize the initialization of UARTO，the code is below：

```
/**********************************Function Info ***********************
** (Function Name):                    InitUart0
**
** (Description):                  uart0(Init UART0)
**
**
** (Input Variable):               void
** (Return Value):                     void
**
** (Macro or constant used):  None
** (Global variable used):     None
**
** (Other function called):        None
**
** (Author):                           robot
** (Date created):                 2010-3-25
**
** (Revised):
** (Date Revised):
**********************************************************************/
void InitUart0(void)
{
        /*（Set baud rate）*/
        UCSR0A = 0x02;                  //  （Set to double velocity mode）
        UBRR0H = 0;


        // (Set main clock to 16M, baud rate to 1M)
        UBRR0L = 1;


        // (Enable UART0’s receiver and transmitter)
        UCSR0B = (1<<RXEN)|(1<<TXEN);


// (Set data frame:8 data bits,1 stop bit)
        UCSR0C = (3<<UCSZ0);
        /*  设置端口状态(Set port state) */
        DDRE &= ~_BV(PE0);  // (Set default direction of RX to input mode)
        PORTE &= ~_BV(PE0);
                            // (Set default state of RX to tri-state)
        DDRE |= _BV(PE1);   // (Set default direction of TX to output mode)
```

```
    PORTE |= _BV(PE1);    // (Set default state of RX to high state)
    DDRA |= _BV(PA0);     // (Enable direction of Port A as output)
    PORTA &= ~_BV(PA0);// (Set Port A state as same to RX)
    DDRA |= _BV(PA1);     // (Set Port A state as same to RX)
    PORTA |= _BV(PA1);
                          // (Set Port A state as same to RX)
}
```

Duplicate and paste the above code to main.c documents，click keep main.c，then click recompile procedure，u can see the compilation result in“Console”child window as below。



```
void InitUart0(void)
{
    /* 设置波特率*/
    UCSR0A = 0x02;          // 设置为倍速模式
    UBRR0H = 0;
    UBRR0L = 1;             // 主时钟频率为16M时，波特率为1M
    /* 接收器与发送器使能*/
    UCSR0B = (1<<RXEN)|(1<<TXEN);
    /* 设置帧格式: 8个数据位, 1个停止位 */
    UCSR0C = (3<<UCSZ0);
    /* 设置端口状态 */
    DDRE &= ~_BV(PE0);      // 初始化RX端口默认方向为输入
    PORTE &= ~_BV(PE0);     // 初始化RX端口默认状态为高阻
    DDRE |= _BV(PE1);       // 初始化TX端口默认方向为输出
    PORTE |= _BV(PE1);      // 初始化TX端口默认状态为高电平

    DDRA |= _BV(PA0);       // 初始化使能端口状态方向为输出
    PORTA &= ~_BV(PA0);     // 初始化使能端口状态方为RX状态
```

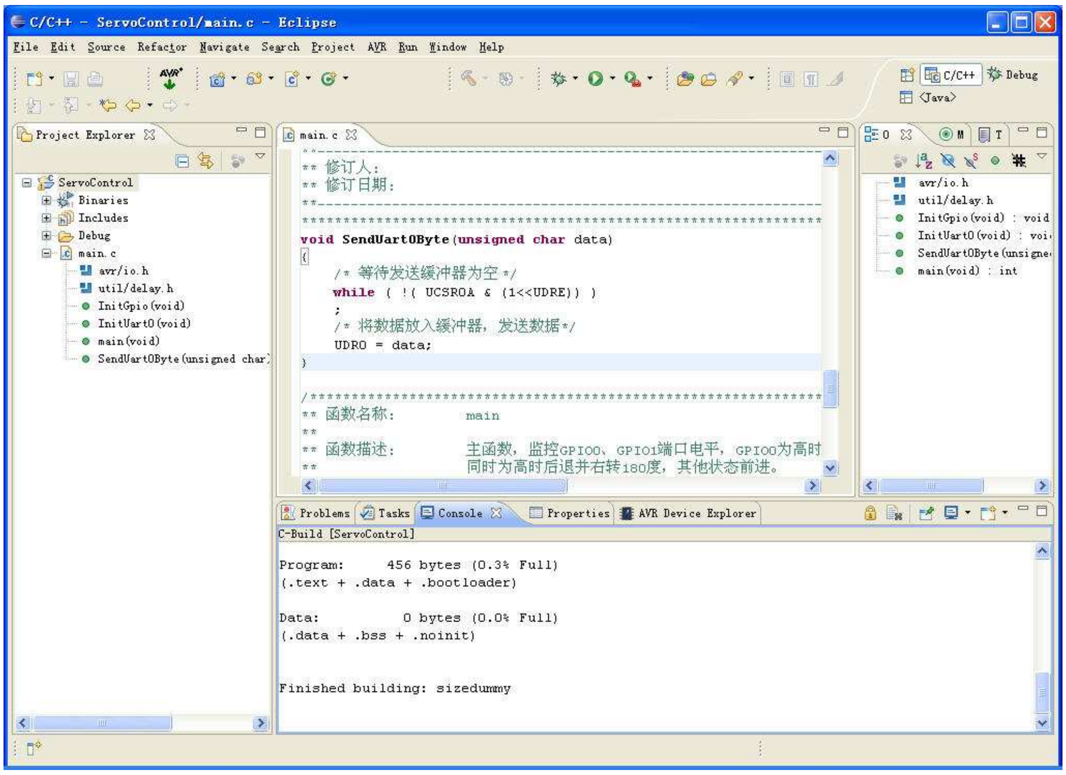
```
C-Build [ServoControl]

Program:     410 bytes (0.3% Full)
(.text + .data + .bootloader)

Data:          0 bytes (0.0% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

5.23 Add servo control function

To realize the control of servo needs to send a control command through UART0 . Control command is usually a frame, the actual data, UART0 is sent by bytes. In order to facilitate the call, first add a function SendUart0Byte in main.c   The HTML code is as follows：

```
/**********************************Function Info ***********************
** (Function Name):                    SendUart0Byte
**
** (Description):                  (Send one byte through UART0)
**
**
** (Input Variable):               unsigned char data
** (Return Value):                      void
**
** (Macro or constant used):        None
** (Global variable used):          None
**
** (Other function called):             None
**
** (Author):                            robot
** (Date created):                      2010-3-25
**
** (Revised):
** (Date Revised):
*************************************************************************/
void SendUart0Byte(unsigned char data)
{

        // waiting for finishing sending all datas in the Transmit Buffer
        while ( !( UCSR0A & (1<<UDRE)) );
        /*Put the data in the buffer then send */
        UDR0 = data;
}
```

Duplicate &paste the above code to main.c documents，click keep main.c，then click recompile procedure，you can see the compilation results in the 在“Console“ child window is shown below。



```
**-------------------------------------------------------------------
** 修订人:
** 修订日期:
**-------------------------------------------------------------------
*********************************************************************
void SendUart0Byte(unsigned char data)
{
    /* 等待发送缓冲器为空 */
    while ( !( UCSR0A & (1<<UDRE)) )
    ;
    /* 将数据放入缓冲器，发送数据*/
    UDR0 = data;
}

/********************************************************************
** 函数名称:          main
**
** 函数描述:          主函数，监控GPIO0、GPIO1端口电平，GPIO0为高时
**                    同时为高时后退并右转180度，其他状态前进。
```

```
C-Build [ServoControl]

Program:     456 bytes (0.3% Full)
(.text + .data + .bootloader)

Data:          0 bytes (0.0% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

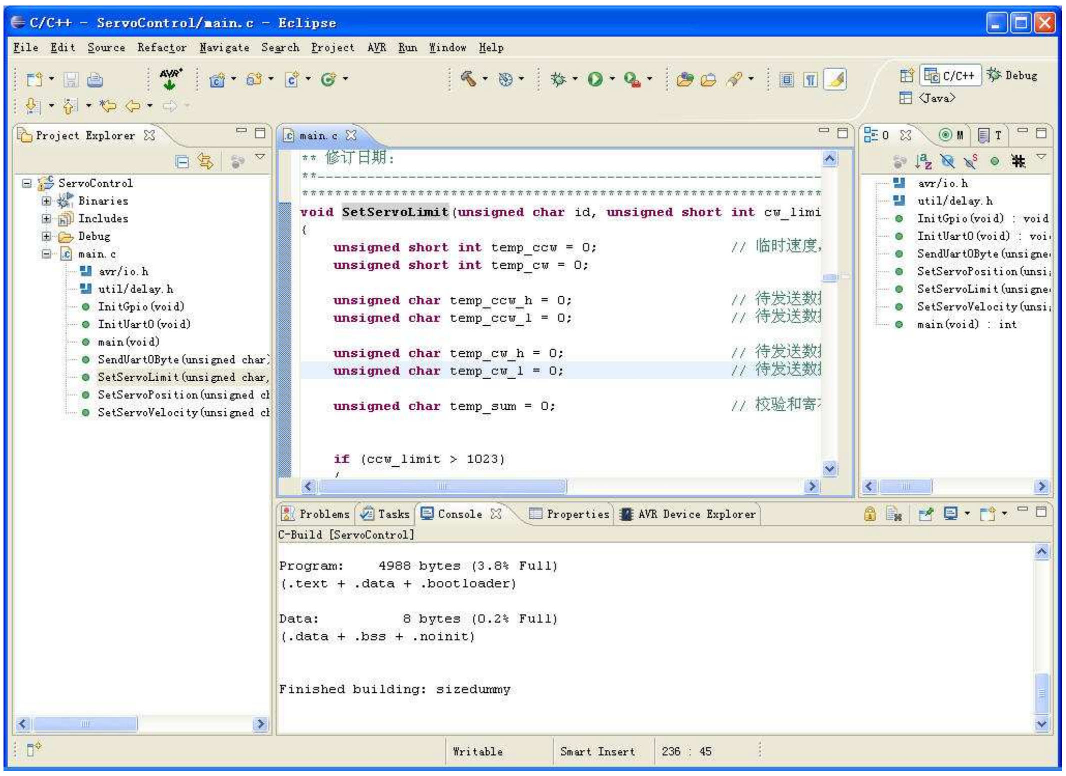
SR518 series servos can be set in two modes: servo mode and motor mode. When servos come out of the factory, it is considered to be the default servo mode. The mode Settings can be realized by setting the minimum and maximum limit parameters. Here add a SetServoLimit function to set the servo mode, when program needs to change the actual operation mode, you can invoke this function, the code is below：

```
/*********************************Function Info ***********************
** (Function Name):                    SetServoLimit
**
** (Description):          (Set servo position limitation and change servo mode)
**
**
** (Input Variable):               unsigned char id; unsigned short int cw_limit;signed short int ccw_limit;
** (Return Value):                     void
**
** (Macro or constant used):       None
** (Global variable used):         None
**
** (Other function called):            None
**
** (Author):                           robot
** (Date created):                     2010-3-25
**
** (Revised):
** (Date Revised):
```

```
*****************************************************************************/
void SetServoLimit(unsigned char id, unsigned short int cw_limit, unsigned short int ccw_limit)
{
        unsigned short int temp_ccw = 0;                                                              //   (temp
velocity to judge the direction)
        unsigned short int temp_cw = 0;

        unsigned char temp_ccw_h = 0;           // (h bits to be sended)
        unsigned char temp_ccw_l = 0;           // (l bits to be sended)
        unsigned char temp_cw_h = 0;
        unsigned char temp_cw_l = 0;

        unsigned char temp_sum = 0;                     // (temp variable to save checksum)
        if (ccw_limit > 1023)
        {
                temp_ccw = 1023;                        // (limit the velocity to 0-1023)
        }
        else
        {
                temp_ccw = ccw_limit;
        }

        if (cw_limit > 1023)
        {
                temp_cw = 1023;
        }
        else
        {
                temp_cw = cw_limit;
        }

        temp_ccw_h = (unsigned char)(temp_ccw >> 8);
        temp_ccw_l = (unsigned char)temp_ccw;
                // (split 16 bits to 2 bytes)
        temp_cw_h = (unsigned char)(temp_cw >> 8);
        temp_cw_l = (unsigned char)temp_cw;
                // (split 16 bits to 2 bytes)
        PORTA &= ~_BV(PA1);
        PORTA |= _BV(PA0);
                // (Set the bus to host transmit state)
        UCSR0A |= (1<<TXC0);
                        // (Clear written flag of UART0)
        SendUart0Byte(0xFF);
                // (Send the start byte 0xff)
```

```
    SendUart0Byte(0xFF);
                // (Send the start byte 0xff)
    SendUart0Byte(id);                    //  发送 id
                                          // (Send the servo’s ID)
    SendUart0Byte(7);                     3
                                          // (Send the length of frame)
    SendUart0Byte(0x03);          // (Send command “WRITE DATA”)
    SendUart0Byte(0x06);          // (Send the start address of control rigister)
    SendUart0Byte(temp_cw_l);  // (Send the low byte of clockwise position limit)
    SendUart0Byte(temp_cw_h);
                                          //(Send the high byte of clockwise position limit)
    SendUart0Byte(temp_ccw_l);  // (Send the low byte of counterclockwise position limit)
    SendUart0Byte(temp_ccw_h);      // (Send the low byte of counterclockwise position limit)
    temp_sum = id + 7 + 0x03 + 0x06 + temp_cw_l + temp_cw_h + temp_ccw_l + temp_ccw_h;
    temp_sum = ~temp_sum;             // (Calculate the checksum)
    SendUart0Byte(temp_sum);   // (Send checksum)

    while ( !( UCSR0A & (1<<TXC0)) )
    {                                     // (Waiting for finishing sending)
        ;
    }

    PORTA |= _BV(PA1);
    PORTA &= ~_BV(PA0);         // (Set the UART bus to host receiving state)
    _delay_ms(2);
                        // (The bus will be overrode by slave after finishing sending
                        // to receive the answer, so here delays 2 ms.)
}
```

Duplicate & paste the above code to main.c documents，click keep main.c，then click recompile procedure，u can see the compilation result in "Console"child window as below。


C/C++ - ServoControl/main.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
ServoControl
Binaries
Includes
Debug
main.c
avr/io.h
util/delay.h
InitGpio(void)
InitUart0(void)
main(void)
SendUart0Byte(unsigned char)
SetServoLimit(unsigned char,
SetServoPosition(unsigned ch
SetServoVelocity(unsigned ch
main.c
Problems
Tasks
Console
Properties
AVR Device Explorer
Writable
Smart Insert
236 : 45

```
** 修订日期:
**------------------------------------------------------------------
********************************************************************
void SetServoLimit(unsigned char id, unsigned short int cw_limi
{
    unsigned short int temp_ccw = 0;                  // 临时速度,
    unsigned short int temp_cw = 0;

    unsigned char temp_ccw_h = 0;                     // 待发送数
    unsigned char temp_ccw_l = 0;                     // 待发送数

    unsigned char temp_cw_h = 0;                      // 待发送数
    unsigned char temp_cw_l = 0;                      // 待发送数

    unsigned char temp_sum = 0;                       // 校验和寄


    if (ccw_limit > 1023)
```

```
C-Build [ServoControl]

Program:    4988 bytes (3.8% Full)
(.text + .data + .bootloader)

Data:          8 bytes (0.2% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

Then add servo position control function SetServoPosition in main.c，this function will realize the sending servo position control command through UART0,code is below：

/*********************************Function Info ***********************

** (Function Name): SetServoPosition

**

** (Description): (Set servo postion)

**

**

** (Input Variable): unsigned char id; unsigned short int position; signed short int velocity;

** (Return Value): void

**

** (Macro or constant used): None

** (Global variable used): None

**

** (Other function called): None

**

** (Author): robot

** (Date created): 2010-3-25

**

** (Revised):

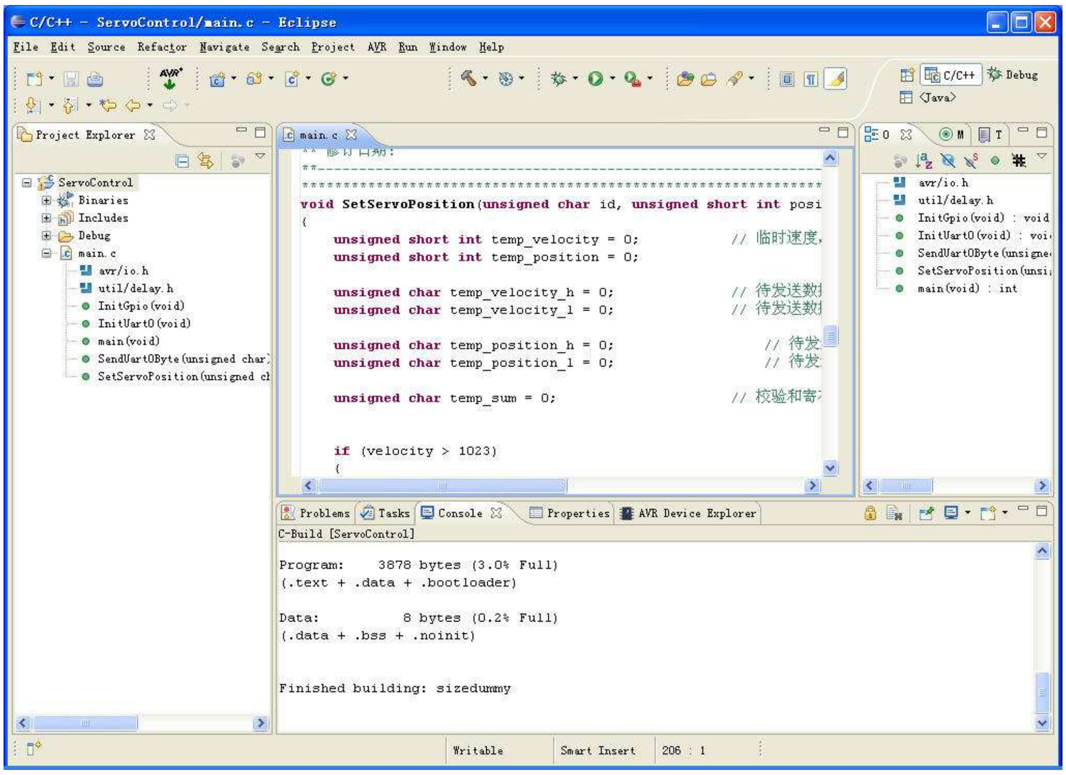
** (Date Revised):

**/

void SetServoPosition(unsigned char id, unsigned short int position, unsigned short int velocity)

```
{
        unsigned short int temp_velocity = 0;
        // (temp velocity to judge the direction)
        unsigned short int temp_position = 0;

        unsigned char temp_velocity_h = 0;
        // (h bits to be sended)
        unsigned char temp_velocity_l = 0;
                                                                        // (l bits to be sended)
        unsigned char temp_position_h = 0;
        unsigned char temp_position_l = 0;

        unsigned char temp_sum = 0;
        // (temp variable to save checksum)

        if (velocity > 1023)
        {
                temp_velocity = 1023;
        // (limit the velocity to 0-1023)
        }
        else
        {
                temp_velocity = velocity;
        }

        if (position > 1023)
        {
                temp_position = 1023;
        }
        else
        {
                temp_position = position;
        }

        temp_velocity_h = (unsigned char)(temp_velocity >> 8);

        // (split 16 bits to 2 bytes)
        temp_velocity_l = (unsigned char)temp_velocity;

        temp_position_h = (unsigned char)(temp_position >> 8);
                // (split 16 bits to 2 bytes)
        temp_position_l = (unsigned char)temp_position;

        PORTA &= ~_BV(PA1);
        PORTA |= _BV(PA0);                        // (Set the bus to host transmit state)
        UCSR0A |= (1<<TXC0);                      // (Clear written flag of UART0)
        SendUart0Byte(0xFF);
                                               // (Send the start byte 0xff)
        SendUart0Byte(0xFF);

        SendUart0Byte(id);                                // (Send the servo’s ID)
        SendUart0Byte(7);                      // Send data length is parameter length+2,parameter
length is 3
                                               // (Send the length of frame)
        SendUart0Byte(0x03);                   // (Send command “WRITE DATA”)
        SendUart0Byte(0x1E);
                                               // (Send the start address of control register)
        SendUart0Byte(temp_position_l);        // (Send the low byte of velocity)
        SendUart0Byte(temp_position_h);        // (Send the high byte of velocity)
        SendUart0Byte(temp_velocity_l);        // (Send the low byte of position)
        SendUart0Byte(temp_velocity_h);
                                               // (Send the high byte of position)
        temp_sum = id + 7 + 0x03 + 0x1E + temp_position_l + temp_position_h + temp_velocity_l +
temp_velocity_h;
        temp_sum = ~temp_sum;                     // (calculate the checksum)
        SendUart0Byte(temp_sum);               // (Send the checksum)

        while ( !( UCSR0A & (1<<TXC0)) )          {                                        //
(Waiting for finishing sending)
            ;
        }

        PORTA |= _BV(PA1);
        PORTA &= ~_BV(PA0);
                                          // (Set the UART bus to host receiving state)
        _delay_ms(2);
                            // (The bus will be overrode by slave after finishing sending
                            // to receive the answer, so here delays 2 ms.)

}
```

Duplicate & paste the above code to main.c documents,click keep main.c， then click recompile procedure，u can see the compilation result in“Console“child window as below。



```
** 修订日期:
**--------------------------------------------------------------------------
****************************************************************************
void SetServoPosition(unsigned char id, unsigned short int posi
{
    unsigned short int temp_velocity = 0;          // 临时速度,
    unsigned short int temp_position = 0;

    unsigned char temp_velocity_h = 0;             // 待发送数
    unsigned char temp_velocity_l = 0;             // 待发送数

    unsigned char temp_position_h = 0;                 // 待发
    unsigned char temp_position_l = 0;                 // 待发

    unsigned char temp_sum = 0;                    // 校验和寄


    if (velocity > 1023)
    {
```

```
C-Build [ServoControl]

Program:    3878 bytes (3.0% Full)
(.text + .data + .bootloader)

Data:          8 bytes (0.2% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

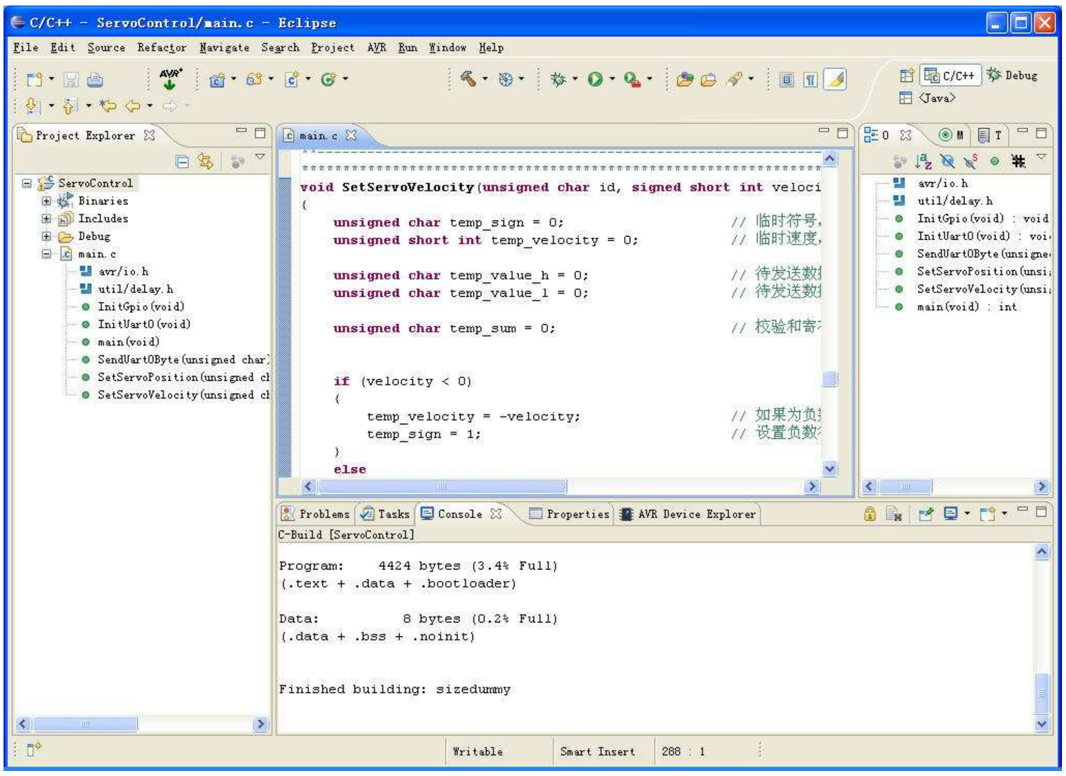
For motor mode,when u control u just need to set the servo's speed。Here add the function SetServoVelocity to realize the sending speed control command through UART0,the code is below：

```
/***********************************Function Info ***********************
** (Function Name):                    SetServoVelocity
**
** (Description):              (Set servo velocity)
**
**
** (Input Variable):               unsigned char id; signed short int velocity;
** (Return Value):                     void
**
** (Macro or constant used):       None
** (Global variable used):         None
**
** (Other function called):            None
**
** (Author):                           robot
** (Date created):                     2010-3-25
**
** (Revised):
** (Date Revised):
***************************************************************************/
void SetServoVelocity(unsigned char id, signed short int velocity)
{
```

```
unsigned char temp_sign = 0;
// (temp variable to judge the direction)
unsigned short int temp_velocity = 0;
// (temp velocity to judge the direction)
unsigned char temp_value_h = 0;
// (h bits to send)
unsigned char temp_value_l = 0;
// (l bits to send)
unsigned char temp_sum = 0;
// (temp variable to save checksum)

if (velocity < 0)
{
        temp_velocity = -velocity;
// (if negative, get the absolute value)
        temp_sign = 1;
                                                    // (Set negative flag)
}
else
{
        temp_velocity = velocity;
        temp_sign = 0;
        // (Set positive flag)
}

if (temp_velocity > 1023)
{
        temp_velocity = 1023;
// (Limit the velocity to 0-1023)
}

// (Set bit0 as direction bit, then temp_velocity is the data to be sended)
temp_velocity |= (temp_sign << 10);

temp_value_h = (unsigned char)(temp_velocity >> 8);
        // (Split the 16 bits to 2 bytes)
temp_value_l = (unsigned char)temp_velocity;

PORTA &= ~_BV(PA1);
PORTA |= _BV(PA0);                          // (Set the bus to host transmit state)
UCSR0A |= (1<<TXC0);                        // (Clear written flag of UART0)
SendUart0Byte(0xFF);
                                      // (Send the start byte 0xFF)
SendUart0Byte(0xFF);              // (Send the start byte 0xFF)
```

```
        SendUart0Byte(id);                          // (Send the servo’s ID)
        SendUart0Byte(5);
                                                    // (Send the length of frame)
        SendUart0Byte(0x03);
                                                    // (Send command “WRITE DATA”)
        SendUart0Byte(0x20);                 // (Send the start address of control register)
        SendUart0Byte(temp_value_l);    // (Send the low byte of velocity)
        SendUart0Byte(temp_value_h);
                                                    // (Send the high byte of velocity)
        temp_sum = id + 5 + 0x03 + 0x20 + temp_value_l + temp_value_h;
        temp_sum = ~temp_sum;                            // (Calculate the checksum)
        SendUart0Byte(temp_sum);                    // (Send the checksum)
        while ( !( UCSR0A & (1<<TXC0)) )                   {                                              //
(Waiting for finishing sending)
                ;
        }

        PORTA |= _BV(PA1);
        PORTA &= ~_BV(PA0);// (Set the UART bus to host receiving state)
        _delay_ms(2);
                                //the bus will be overrode by slave after finishing sending
                                // to receive the answer, so here delays 2 ms.
}
```

Duplicate & paste the abovecode to main.c documents, click keep main.c, then click recompile procedure, u can see the compilation result in the 在“Console“ child window as below。



```
*********************************************************************
void SetServoVelocity(unsigned char id, signed short int veloci
{
    unsigned char temp_sign = 0;                      // 临时符号,
    unsigned short int temp_velocity = 0;             // 临时速度,

    unsigned char temp_value_h = 0;                   // 待发送数
    unsigned char temp_value_l = 0;                   // 待发送数

    unsigned char temp_sum = 0;                       // 校验和寄


    if (velocity < 0)
    {
        temp_velocity = -velocity;                    // 如果为负
        temp_sign = 1;                                // 设置负数
    }
    else
```

```
C-Build [ServoControl]

Program:    4424 bytes (3.4% Full)
(.text + .data + .bootloader)

Data:          8 bytes (0.2% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

5.24 Realize control logic

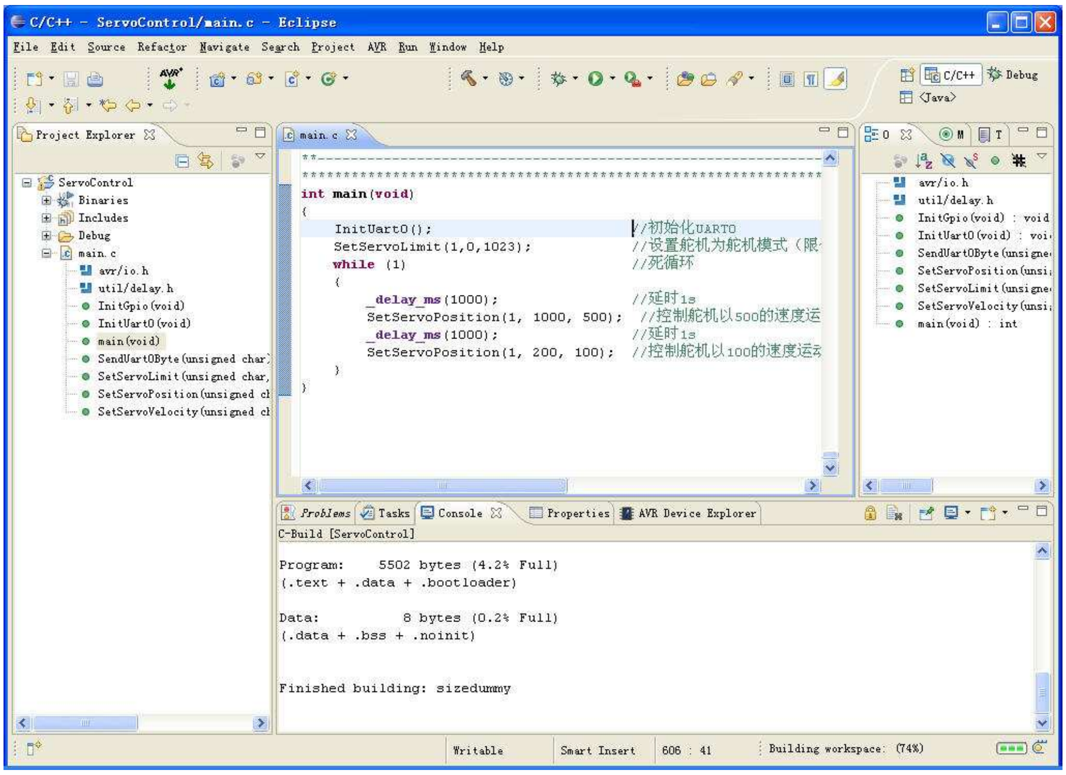
- **The realization of servo swing logic**

When we have servo position and speed control function，then we can complete the control logic of a servo swaying between two positions。Here suppose the ID No. is 1 for the servo needs control，the modified main function code is below：

```
/*********************************Function Info ***********************
** (Function Name):                    main
**
** (Description):                (Wave the Servo of ID 1)
**
**
** (Input Variable):             void
** (Return Value):                    int
**
** (Macro or constant used):     None
** (Global variable used):       None
**
** (Other function called):           None
**
** (Author):                          robot
** (Date created):                    2010-3-25
**
** (Revised):
```

```
** (Date Revised):
**************************************************************************
int main(void)
{
        InitUart0();                                                            // (Init UART0)
  SetServoLimit(1,0,1023);                    //set the servo in servo mode(limit0-1023）
                                                        // (Set servo of ID 1 to servo mode)
        while (1)                                       // (Infinite loop)
        {
                _delay_ms(1000);
                                                        //(Delay 1 s)
                SetServoPosition(1, 1000, 500);                                                 //
(Control the servo of ID 1 to position 1000
                                                        //with the velocity of 500)
                _delay_ms(1000);                                                                //(Delay
1 s)
                SetServoPosition(1, 200, 100);// (Control the servo of ID 1 to position 200
                                                        //with thevelocity of 100)
        }
}
```

Duplicate & paste the above code to the main.c documents，click keep main.c，then click recompile procedure，u can see the compilation result in "Console"child window as below。After right compilation,，we can use thewayof ISP or IAPto download the procedure to the control card.Please refre to the detailed methodlogy《EclipseForAVR program development》。


C/C++ - ServoControl/main.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
ServoControl
Binaries
Includes
Debug
main.c
avr/io.h
util/delay.h
InitGpio(void)
InitUart0(void)
main(void)
SendUart0Byte(unsigned char)
SetServoLimit(unsigned char,
SetServoPosition(unsigned ch
SetServoVelocity(unsigned ch
main.c
Problems
Tasks
Console
Properties
AVR Device Explorer
Writable
Smart Insert
606 : 41
Building workspace: (74%)

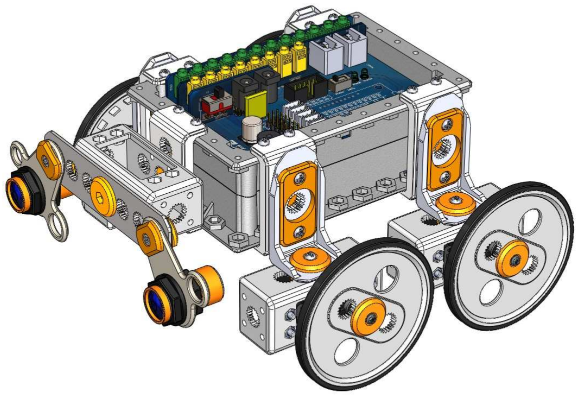
```
**---------------------------------------------------------------
*****************************************************************
int main(void)
{
    InitUart0();                        //初始化UART0
    SetServoLimit(1,0,1023);            //设置舵机为舵机模式（限
    while (1)                           //死循环
    {
        _delay_ms(1000);                //延时1s
        SetServoPosition(1, 1000, 500);  //控制舵机以500的速度运
        _delay_ms(1000);                //延时1s
        SetServoPosition(1, 200, 100);  //控制舵机以100的速度运
    }
}
```

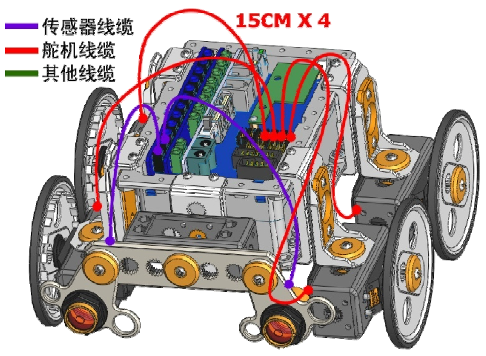
```
C-Build [ServoControl]

Program:    5502 bytes (4.2% Full)
(.text + .data + .bootloader)

Data:          8 bytes (0.2% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

How to achieve the obstacle-avoiding car control logic?

- **The logic realization of obstacle-avoiding car**

In order to debug the program,users can put up/construct a obstacle-avoidingcarrefering to the following picture.The servos’ID on car’s right are 1（front）and 2（back），the servos’ ID on car’s left are 3(front)and 4(back)。The infrared sensor on right must connect to the controller’s IO0，the infrared sensor on left must connect to the controller’s IO1。





To realize the obstacle avoidance, you need to check the sensor to judge whether the car meet barriers or not, so need to add the code to realize sensor data's reading According to the logic of obstacle avoidance, when car hits obstacles.it needs to turn right, left,or movebackwards, no obstacles,need to move forward. In order to make the logic clear, we can add four functions of left, right, forward and backward to invoke

The function GetGpio code of read sensor data is below：

```
/************************************Function Info ***********************
** (Function Name):                GetGpio
**
```

```
** (Description):                (Get general digital I/O port value)
**
**
** (Input Variable):                unsigned char* val
** (Return Value):                      void
**
** (Macro or constant used):        None
** (Global variable used):          None
**
** (Other function called):             None
**
** (Author):                            robot
** (Date created):                      2010-3-25
**
** (Revised):
** (Date Revised):
***********************************************************************/
void GetGpio(unsigned char* val)
{
    unsigned char temp_input_value = 0;

    temp_input_value = 0;

    if (PINE & _BV(PE5))                // (Get value of PE5)
    {
        temp_input_value |= _BV(0);
                                        // (_BV is a macro to shift, _BV(0) means (1<<0)
    }
    else
    {
        temp_input_value &= ~_BV(0);
    }

    if (PINE & _BV(PE4))                // (Get value of PE5)
    {
        temp_input_value |= _BV(1);
    }
    else
    {
        temp_input_value &= ~_BV(1);
    }

    *val = temp_input_value;
}
```

The code of TernLeft as follows：

```
/*********************************Function Info ***********************
** (Function Name):                    TurnLeft
**
** (Description):                ( Turnleft)
**
**
** (Input Variable):             unsigned short int time,
** (Return Value):                      void
**
** (Macro or constant used):     None
** (Global variable used):       None
**
** (Other function called):             None
**
** (Author):                            robot
** (Date created):                      2010-3-25
**
** (Revised):
** (Date Revised):
******************************************************************************/
void TurnLeft(unsigned short int time)
{
        SetServoVelocity(1, 500);    // (Control the servo of ID 1 to rotate
                                            //with the velocity of 500)
        SetServoVelocity(2, 500);
                                            // (Control the servo of ID 2 to rotate
                                            //with the velocity of 500)
        SetServoVelocity(3, 500);
                                            // (Control the servo of ID 3 to rotate
                                            //with the velocity of 500)
        SetServoVelocity(4, 500);    // (Control the servo of ID 4 to rotate
                                            //with the velocity of 500)
        _delay_ms(time);                    //(Delay time ms)
}
```

The code of TurnRight is below：

```
/*********************************Function Info ***********************
** (Function Name):                    TurnRight
**
** (Description):                (TurnRight)
**
**
** (Input Variable):             unsigned short int time,
** (Return Value):                      void
```

```
**
** (Macro or constant used):        None
** (Global variable used):          None
**
** (Other function called):                None
**
** (Author):                               robot
** (Date created):                         2010-3-25
**
** (Revised):
** (Date Revised):
*********************************************************************************/
void TurnRight(unsigned short int time)
{
        SetServoVelocity(1, -500);
                                           // (Control the servo of ID 1 to rotate
                                           //with the velocity of -500)
        SetServoVelocity(2, -500);
                                           // (Control the servo of ID 2 to rotate
                                           //with the velocity of -500)
        SetServoVelocity(3, -500);    // (Control the servo of ID 3 to rotate
                                           //with the velocity of -500)
        SetServoVelocity(4, -500);    // (Control the servo of ID 4 to rotate
                                           //with the velocity of -500)
        _delay_ms(time);                   //(Delay time ms)
}
```

The code of Forward is below：

```
/**********************************Function Info ***********************
** (Function Name):                        Forward
**
** (Description):                   (Forward)
**
**
** (Input Variable):                unsigned short int time,
** (Return Value):                         void
**
** (Macro or constant used):        None
** (Global variable used):          None
**
** (Other function called):                None
**
** (Author):                               robot
** (Date created):                         2010-3-25
**
```

```
** (Revised):
** (Date Revised):
**************************************************************************/
void Forward(void)
{
        SetServoVelocity(1, 500);      // (Control the servo of ID 1 to rotate
                                               //with the velocity of 500)
        SetServoVelocity(2, 500);      // (Control the servo of ID 2 to rotate
                                               //with the velocity of 500)
        SetServoVelocity(3, -500);     // (Control the servo of ID 3 to rotate
                                               //with the velocity of -500)
        SetServoVelocity(4, -500);
                                               // (Control the servo of ID 4 to rotate
                                               //with the velocity of -500)
}
```

The code of is shown below：

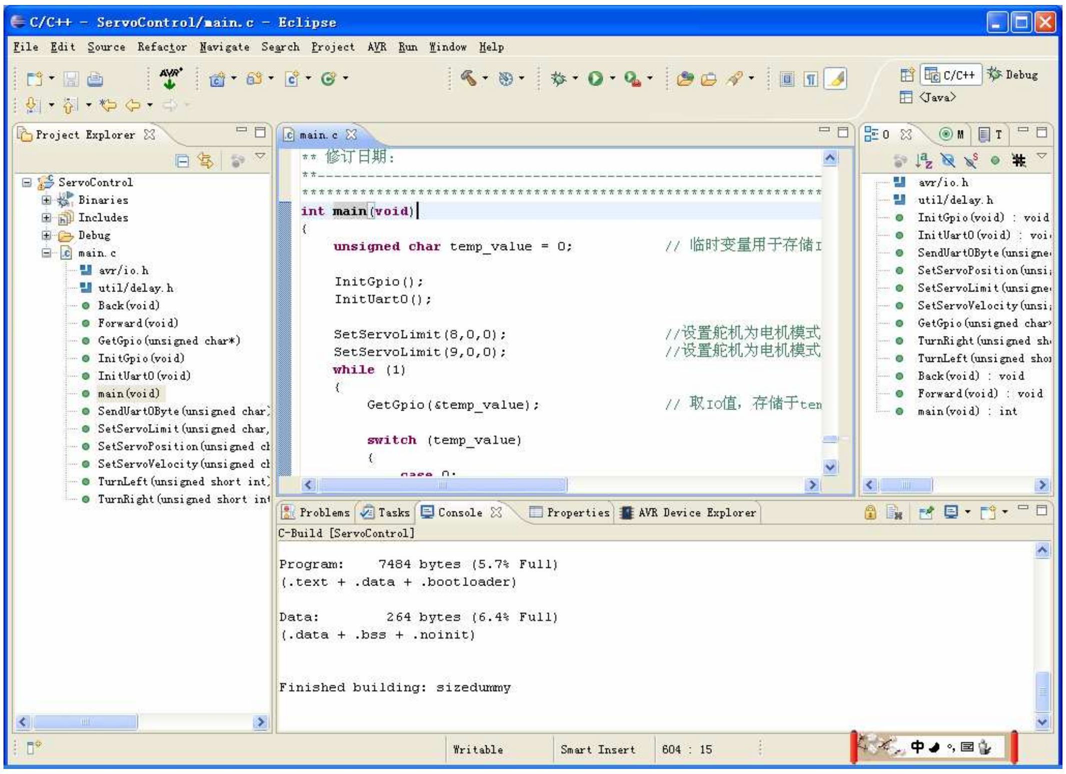
```
/**********************************Function Info ************************
** (Function Name):                        Back
**
** (Description):                  (Back)
**
**
** (Input Variable):               unsigned short int time,
** (Return Value):                          void
**
** (Macro or constant used):       None
** (Global variable used):         None
**
** (Other function called):                 None
**
** (Author):                                robot
** (Date created):                          2010-3-25
**
** (Revised):
** (Date revised):
**************************************************************************/
void Back(void)
{
        SetServoVelocity(1, -500);     // (Control the servo of ID 1 to rotate
                                               //with the velocity of -500)
        SetServoVelocity(2, -500);     // (Control the servo of ID 2 to rotate
                                               //with the velocity of -500)
        SetServoVelocity(3, 500);      // (Control the servo of ID 3 to rotate
                                               //with the velocity of 500)
        SetServoVelocity(4, 500);      // (Control the servo of ID 4 to rotate
                                               //with the velocity of 500)
}
```

Add the function of GetGpio，TurnLeft，TurnRight，Forward and Back to the front main.c function in main 。The code of modified main function is below 如下

```
/***********************************Function Info ***********************
** (Function Name):                       main
**
** (Description):                  ( Logic to control the robot escape
from hit the obstacle)
**
**
** (Input Variable):               void
** (Return Value):                         int
**
** (Macro or constant used):       None
** (Global variable used):         None
**
** (Other function called):                None
**
** (Author):                               robot
** (Date created):                         2010-3-25
**
** (Revised):
** (Date Revised):
*************************************************************************/
int main(void)
{
        unsigned char temp_value = 0;
                                                       // (temp variable to save I/O data)
        InitGpio();
        InitUart0();
        SetServoLimit(1,0,0);
                                                       // (Set the servo of ID 1 to motor mode)
      SetServoLimit(2,0,0);                                                          //
      (Set the servo of ID 2 to motor mode)
          SetServoLimit(3,0,0);                                                    // (Set
          the servo of ID 3 to motor mode)
        SetServoLimit(4,0,0);                                                    //(Set
        the servo of ID 4 to motor mode)

        while (1)
        {
                GetGpio(&temp_value);
```

```
                                                        // (Get the I/O value, save in the
                                                        //bit0 and bit1 of temp_value )
        switch (temp_value)
        {
            case 0:
            {
                Back();                                 //      (Back)
                _delay_ms(2000);                        // 2s (Delay 2 s)
                TurnRight(2000);                        // 2s (Turn right and delay 2 s)
                break;
            }
            case 1:
            {
                Back();
                _delay_ms(1000);
                TurnRight (1000);                       // 1s (Turn right and delay 1 s)
                break;
            }
            case 2:
            {
                Back();
                _delay_ms(1000);
                TurnLeft(1000);
                break;
            }

            default:
            {
                Forward();                                  //
                                                            // (Go forward in other conditions)
                _delay_ms(200);
                break;
            }
        }
    }
}
```

I ate & paste the above code to the main.c documents, click keep main.c，then click restart compiler，in“Console“child window,u can see the compile result as follows。After right compilation， you can use the way of ISP or IAP to download procedures to the control card。Please refer to the detailed methodlogy《EclipseForAVR program development》。

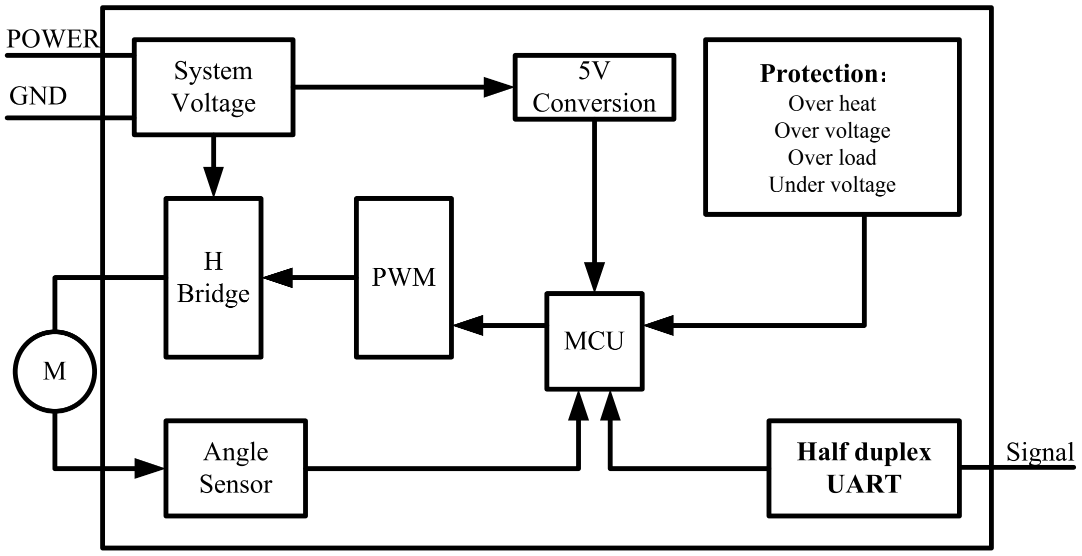

C/C++ - ServoControl/main.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
main.c
Problems Tasks Console Properties AVR Device Explorer
Writable
Smart Insert
604 : 15

```
** 修订日期:
**------------------------------------------------------------------------
*************************************************************************
int main(void)
{
    unsigned char temp_value = 0;            // 临时变量用于存储I

    InitGpio();
    InitUart0();

    SetServoLimit(8,0,0);                    //设置舵机为电机模式
    SetServoLimit(9,0,0);                    //设置舵机为电机模式
    while (1)
    {
        GetGpio(&temp_value);                // 取IO值, 存储于ten

        switch (temp_value)
        {
            case 0:
```

```
C-Build [ServoControl]

Program:    7484 bytes (5.7% Full)
(.text + .data + .bootloader)

Data:        264 bytes (6.4% Full)
(.data + .bss + .noinit)


Finished building: sizedummy
```

Appendix

A. SR518 electric functional block diagram


POWER
GND
System Voltage
5V Conversion
Protection:
Over heat
Over voltage
Over load
Under voltage
H Bridge
PWM
MCU
M
Angle Sensor
Half duplex UART
Signal