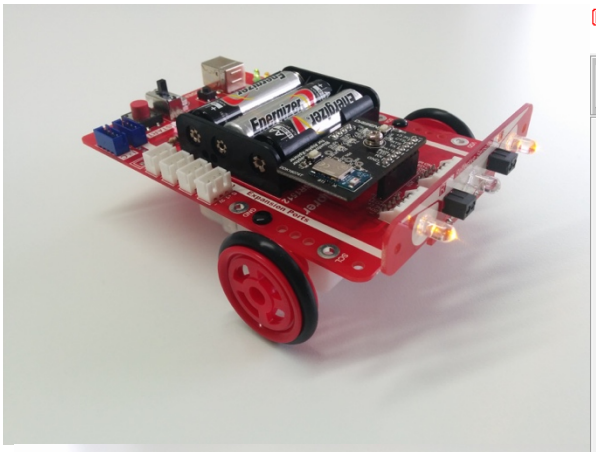
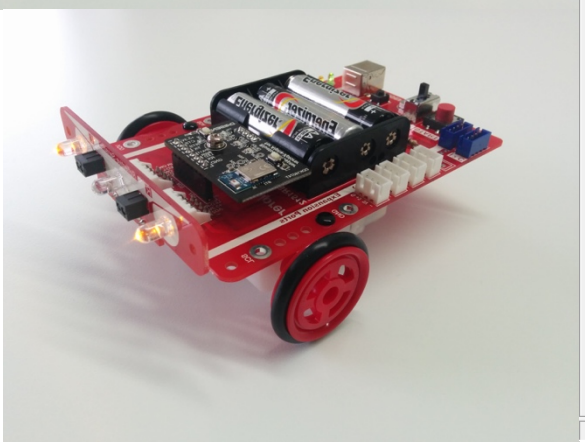
DAISEN

Alpha-Xplorer

Robot Programming Kit

C-Style Block Programming Guide





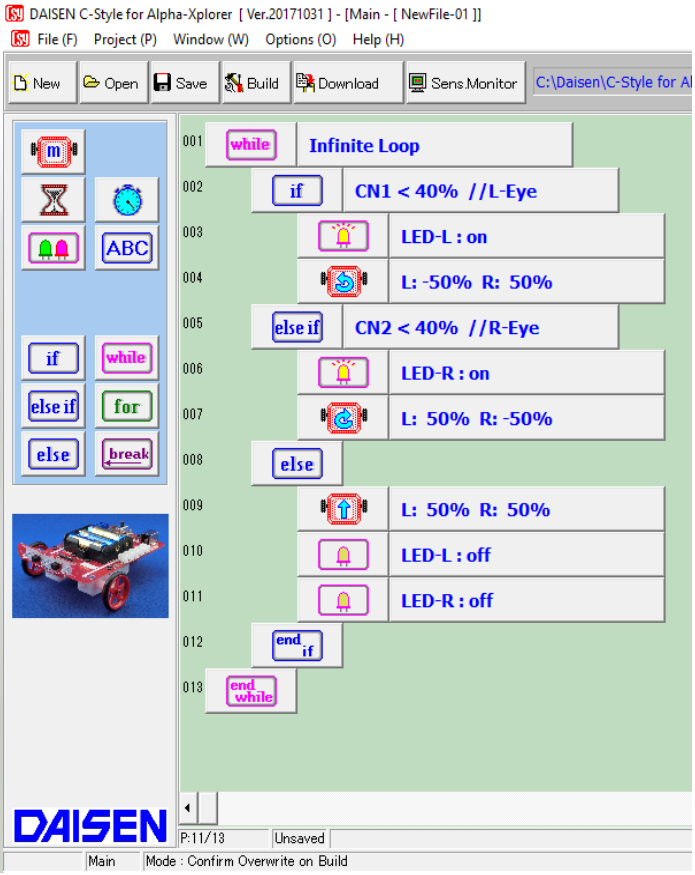


Table of Contents
C-Style Operating Guide

1. Insert, Delete, Copy and Paste Command Buttons 3

1-1. Insert a command button 3
1-2. Delete a command button 4
1-3. Copy and paste command buttons 5

2. Explanation of the Command Buttons 6

2-1. Command button list 6
2-2. Motor control 7
2-3. Wait time 8
2-4. LED control 9
2-5. Timer start 10
2-6. Variables 11
2-7. Condition check – “if”, “else if” and “else” 12
2-8. Repetitive action based on a condition – “while” 13
2-9. Repeat action a set number of times – “for” 14
2-10 Time check 15
2-11 Variable check 16

3. Setting up Input, Output and Extended Functionality 18

3-1. Setup button
3-2. Using a servo motor 21
3-3. Extended functionality 22
3-4. Using the LCD board (DSR1416) 24
3-5. Using the 4 or 6 channel motor control board 25
3-6. Using the sonar distance sensor 26
3-7. Using a compass sensor (6D/9D-Compass: DSR1401/DSR1603) 28
3-8. Connecting several Alpha-Xplorer boards 30

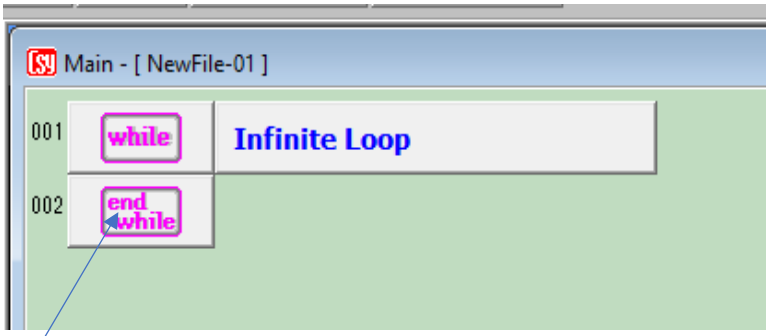
4. Sub-Programs 31

4-1. Finding the sub-program button 31
4-2. Creating sub-programs 32

1. Insert, Delete, Copy and Paste Command Buttons

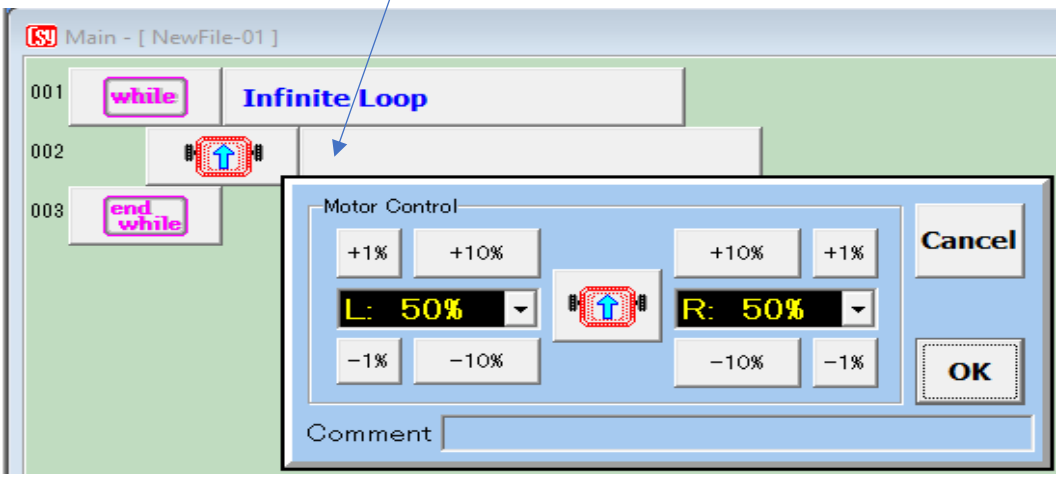
1-1. Insert a command button

1. Select a command you want to use from the command button list and select by clicking on it with your mouse



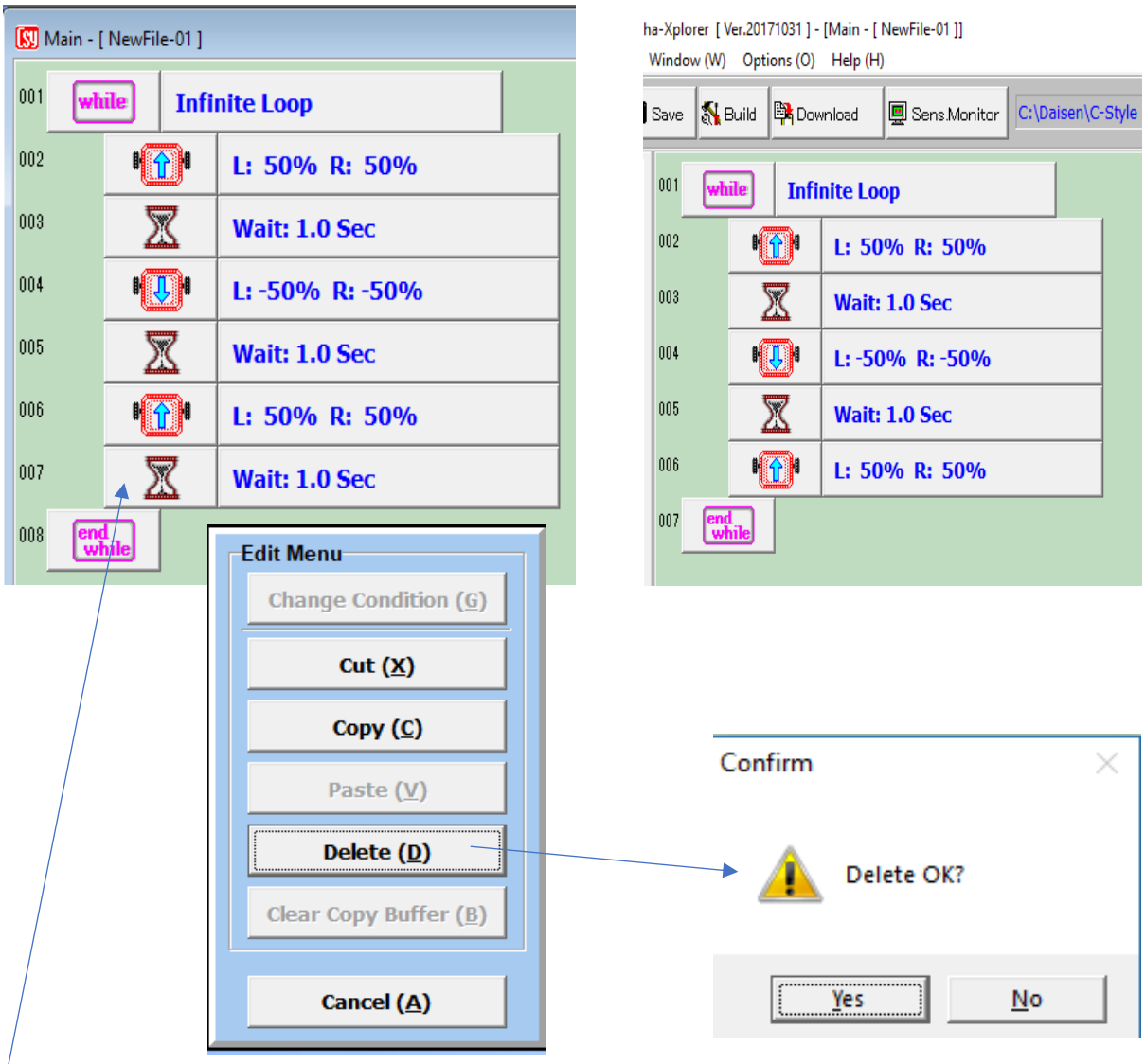
2. Select the position in your program where you want to insert your command and click your mouse there

3. Your new command will be added into your program, and a small dialogue window opens up to set the conditions



In this example, the motor command was selected from the command button list and inserted into the program by clicking on the "end while" button.

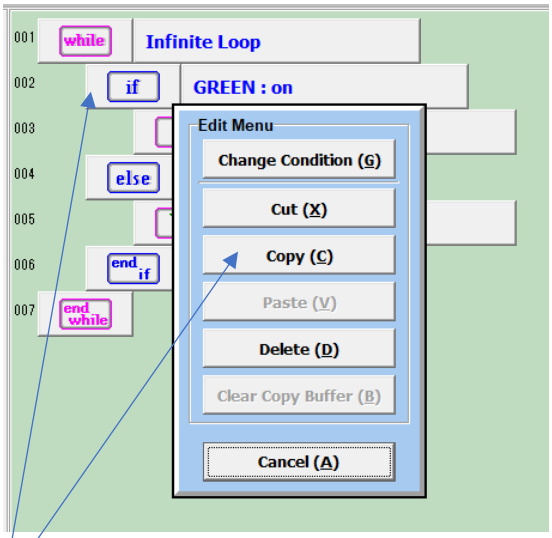
1-2. Delete a command button



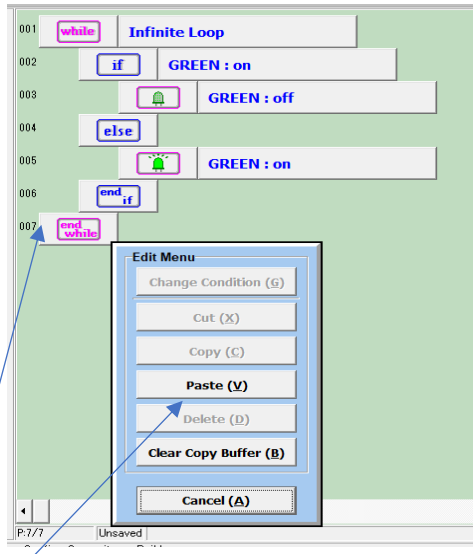
1. Click your mouse on the command button on the left side of the command line you want to delete. A popup menu appears. Click your mouse on “delete”
2. A confirmation window pops up. Click your mouse on “Yes”, and the command line will be deleted from your program.

The popup menu also lets you do other things by clicking on the “Cut”, “Copy” or “Paste” buttons.
When you select the “while”, “for” or “if” command button, all other command until “end while”, “end for” or “end if” will also be deleted (or cut, copied or pasted)

1-3. Copy and paste command buttons

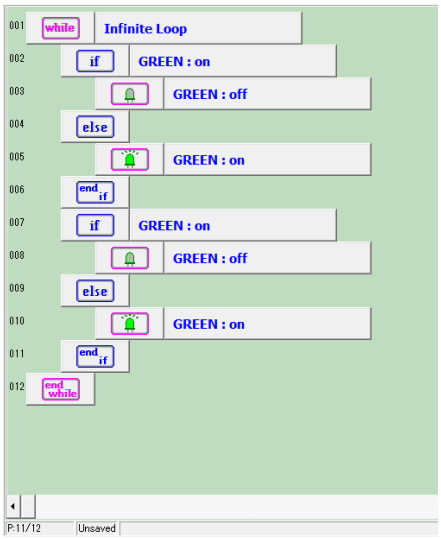


To copy a command or section of code, click your mouse on the command button on the left side of the line, then select “copy” from the pop-up menu.



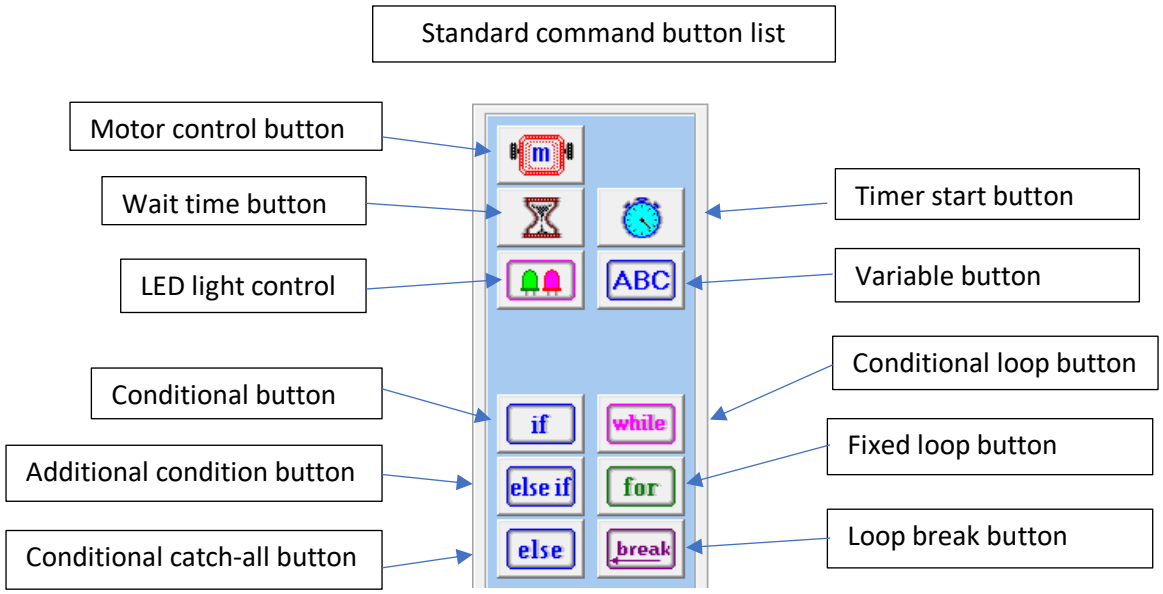
To paste the command or code you copied, click on the command button where you want to insert the code, and select “paste” from the pop-up menu. The code will be inserted.

If you copy several times, the newly copied code will be added behind the code copied first, and when you paste it, all of it will be inserted. “Cut” works the same way. If you don’t need the code any more which you have collected by copying and cutting, select “refresh buffer” from the pop-up menu.

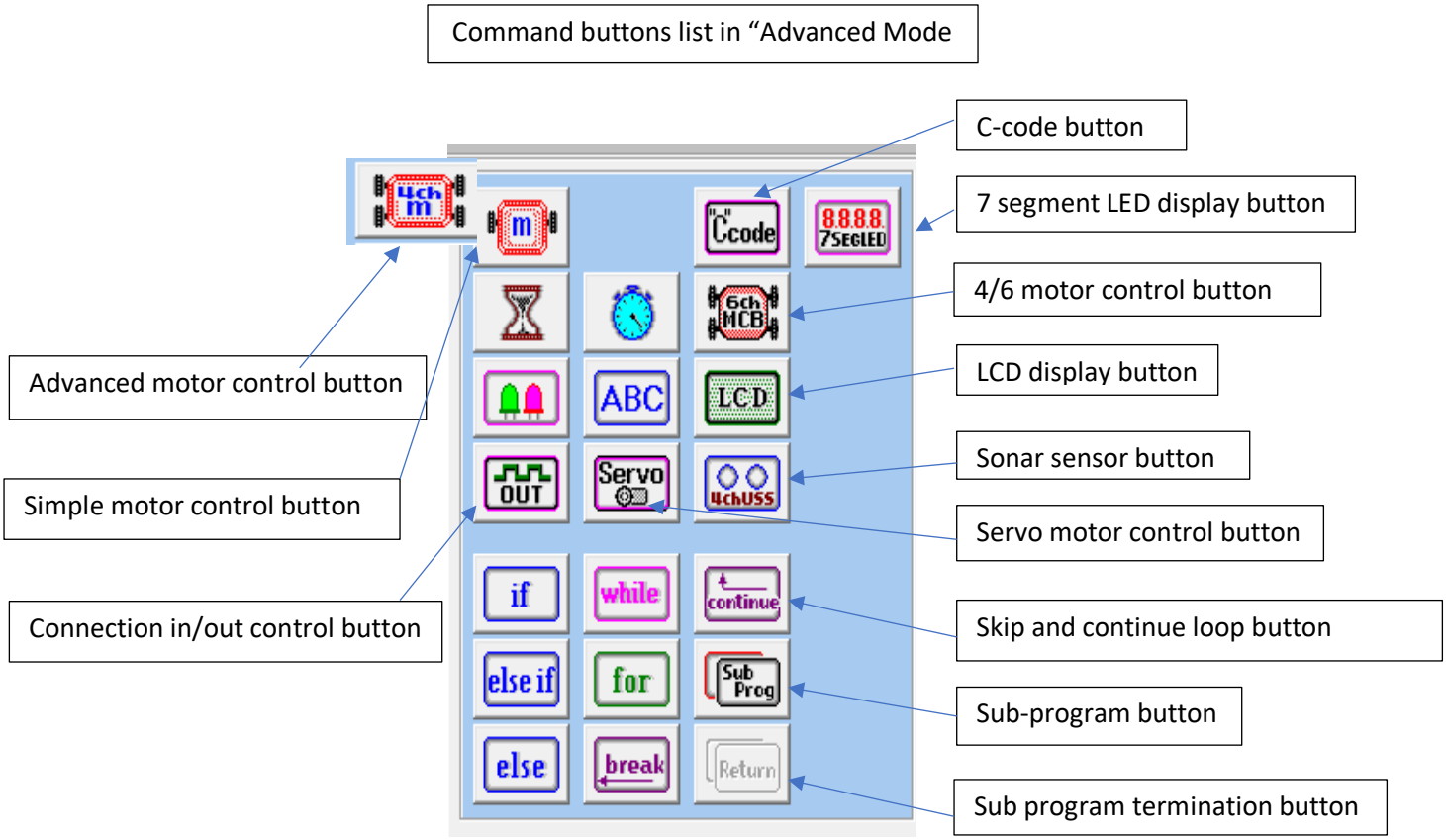


2. Explanation of the Command Buttons

2-1. Command button list

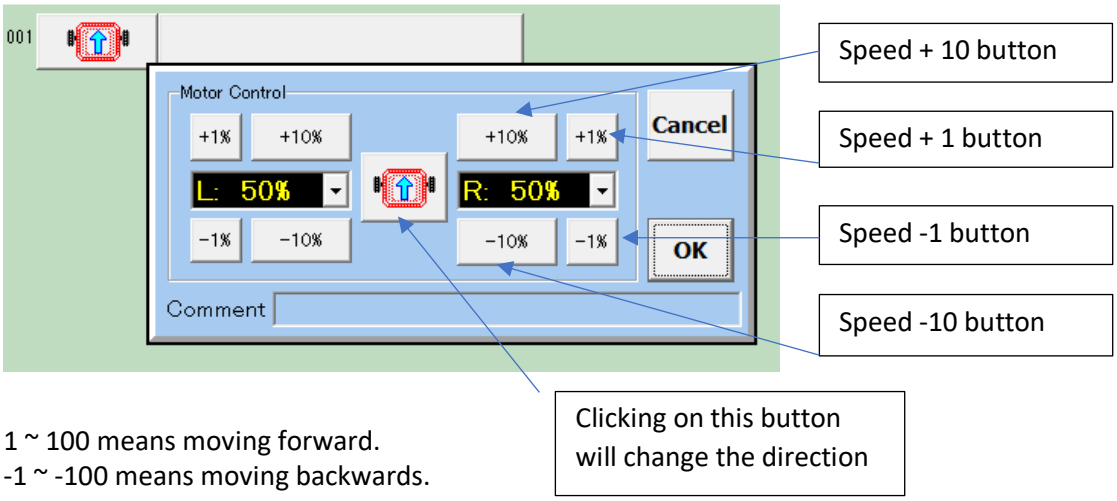


◆Additional command buttons are available in “Advanced Mode”, as explained in chapter 3.



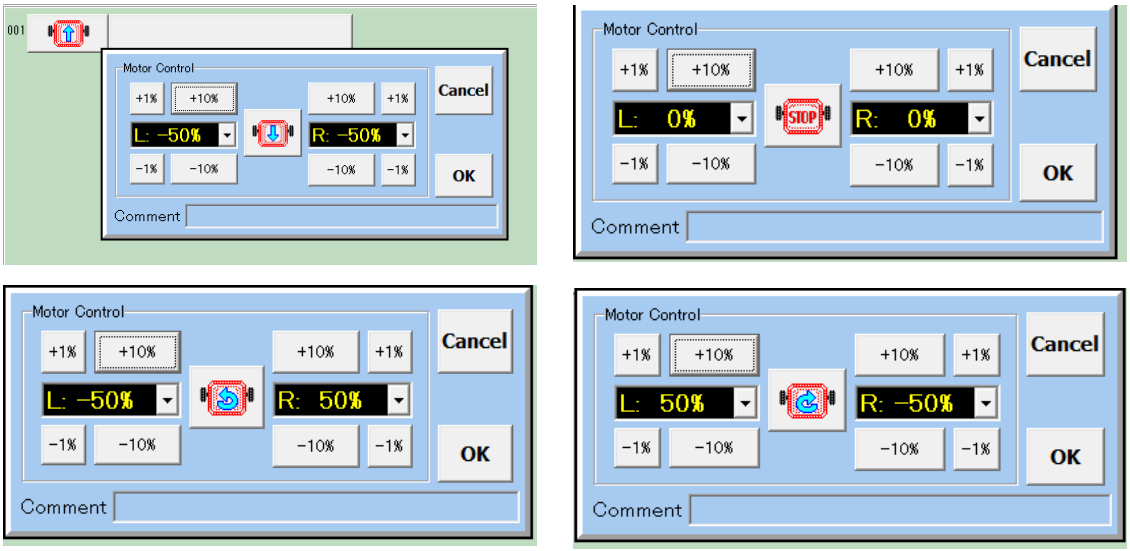
2-2 Motor control

The motor control button sets the motor rotation speed. Therefore, it controls the direction and speed how the robot moves. (L: left motor; R: right motor)



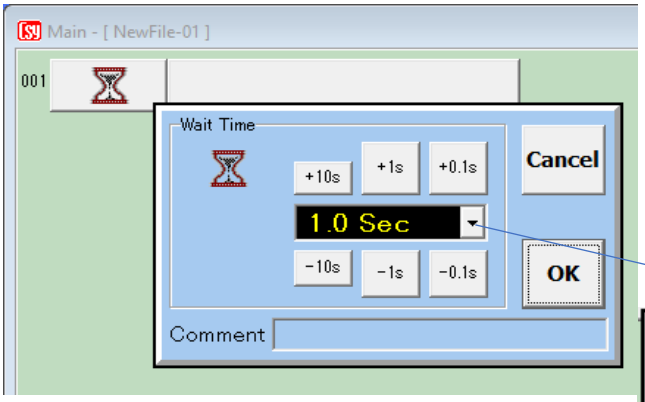
1 ~ 100 means moving forward.
-1 ~ -100 means moving backwards.
0 means the motor stops.

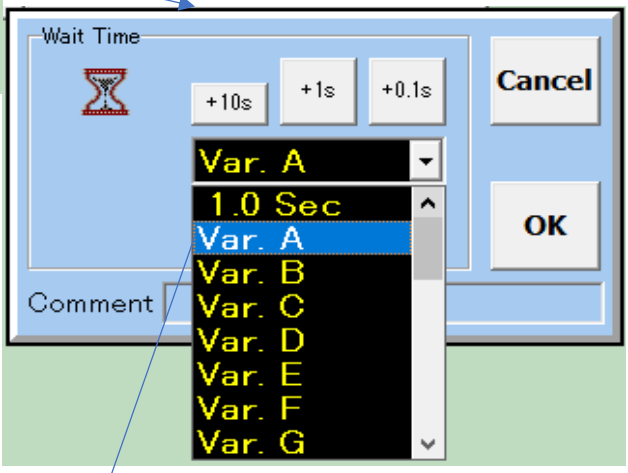
Setting different signs +/- for left and right means the robot will rotate.
Clicking the motor button in the middle several times lets you easily program the robot's movement. It will change the robot's motion from going forward to stop to rotate left to rotate right. After setting speed of the right and left motor, click "OK".
If you adjust either the right or left rotation speed and then click the motor button in the middle, then the other side's rotation speed will be set to the same value.



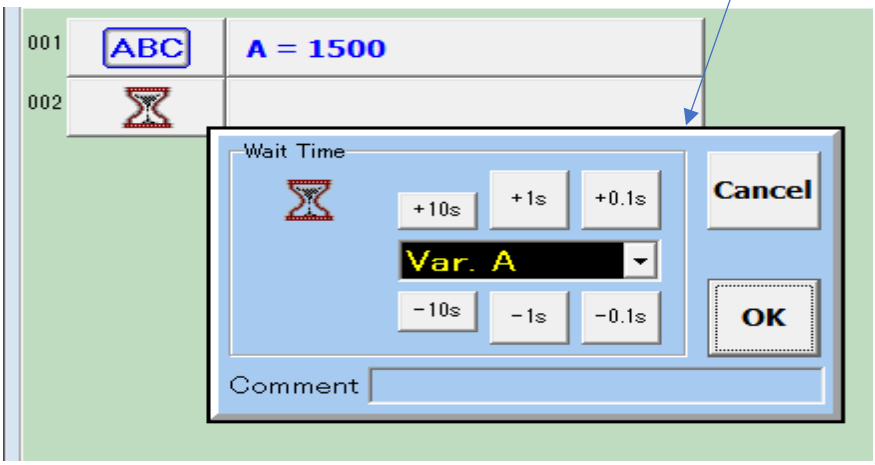
Use the speed change buttons to set the forward or backward speed for each motor.
In addition, you can open the pull-down menu for the speed setting field and set it to a program variable, A-Z. Chapter 2.6 will explain how to work with variables.

2-3 Wait time



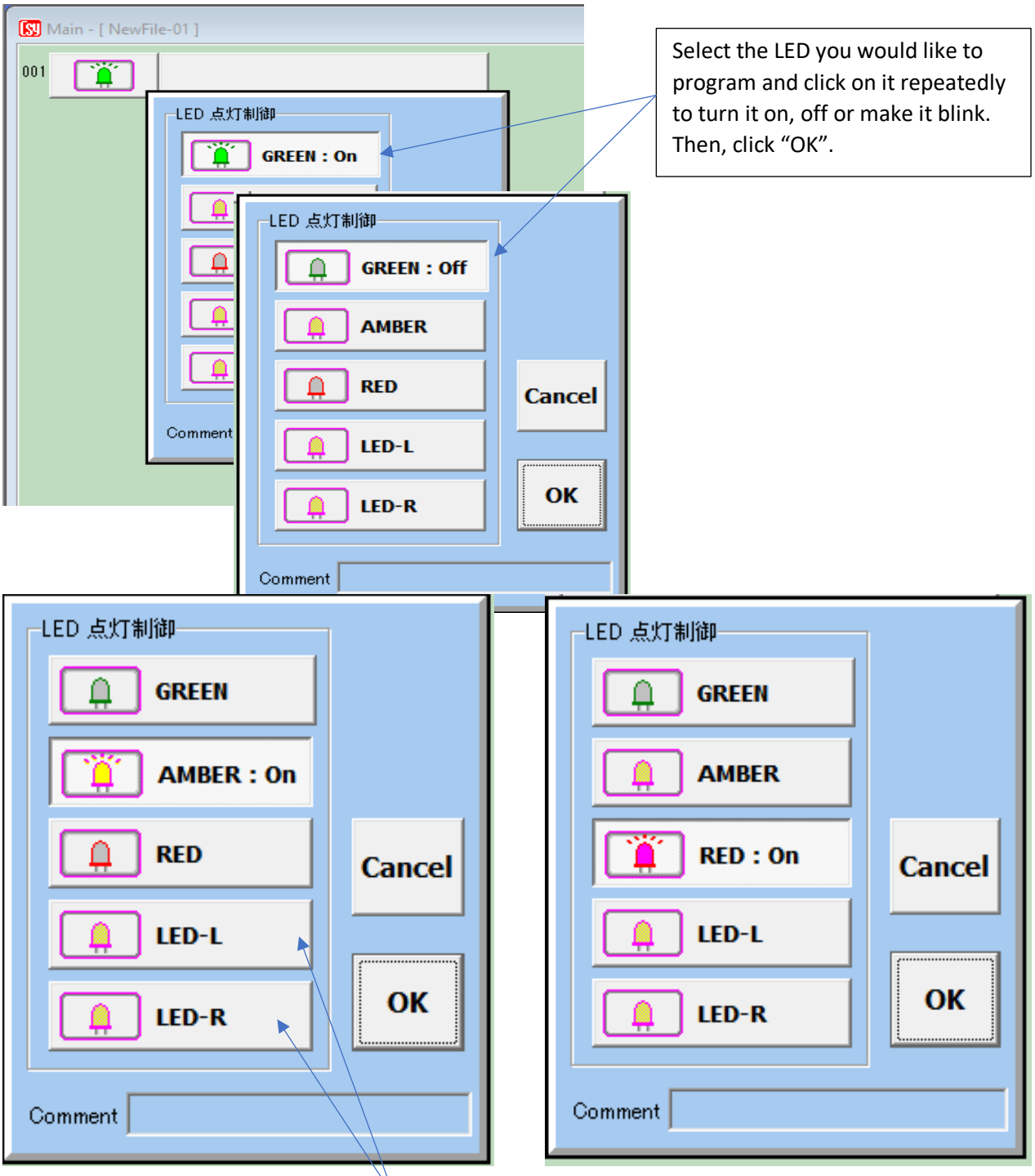


Often, you want your robot to continue doing one thing for some time. In that case, you can make the program wait a while by setting a wait timer. Wait timer can wait for up to 60 seconds, in 0.1 second steps.
If you set the wait time to the value of a program variable, the program will read the variable as milliseconds, so 1000 is one means 1 second. For example, program shown in the last picture sets A to 1500 milliseconds and then waits for 1.5 seconds.



With the wait time button, you can easily program the robot to keep doing the same thing for some time, but during that wait time, you cannot control the robot with other command buttons, for example to react to sensor signals. If you want the robot to be responsive while doing something, set a up a timer and a "while" loop as explained in the following sections of this chapter.

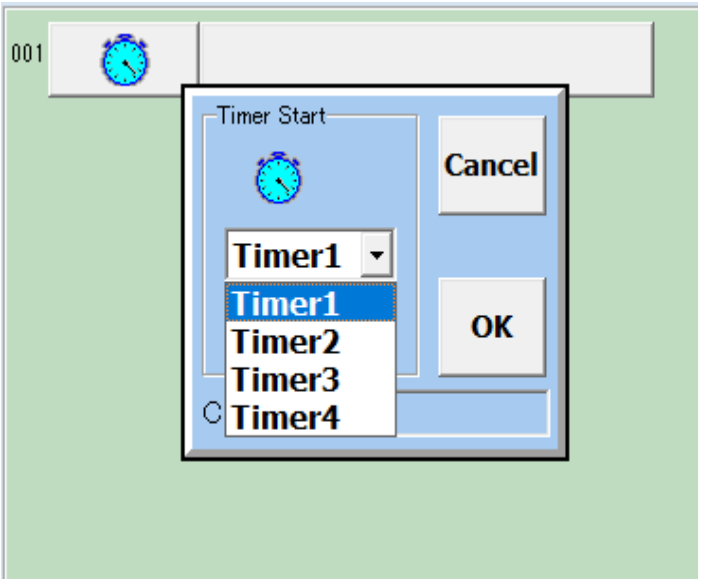
2-4 LED control



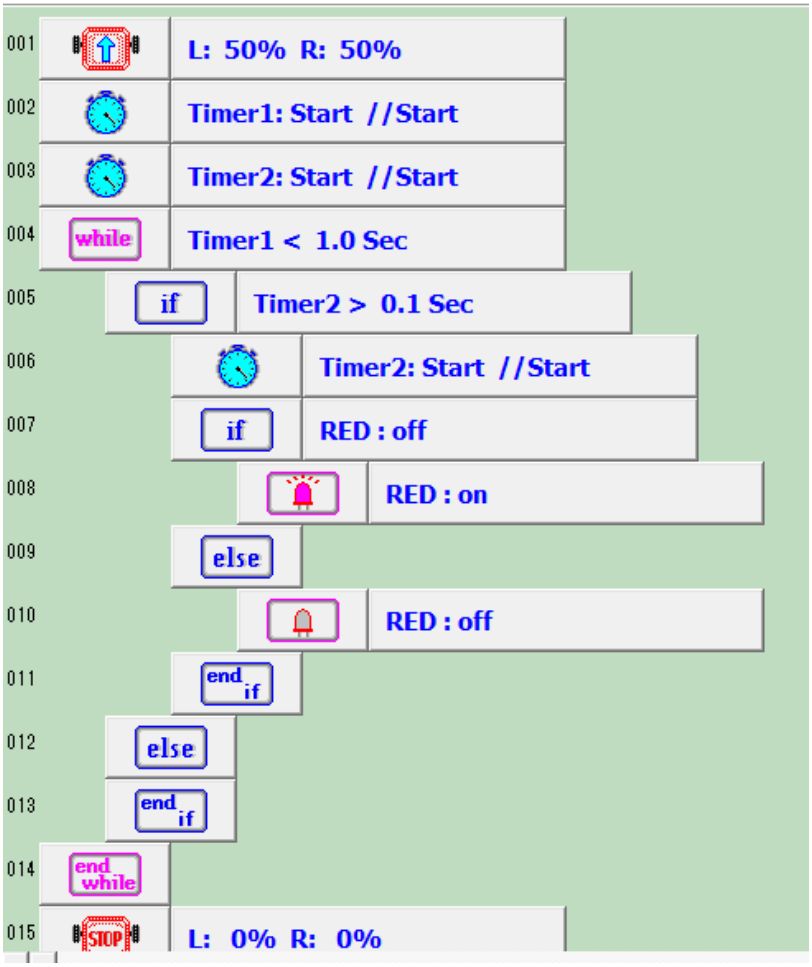
Select the LED you would like to program and click on it repeatedly to turn it on, off or make it blink. Then, click “OK”.

LED-L and LED-R are the LEDs on the right and left side on the sensor board

2-5 Timer start



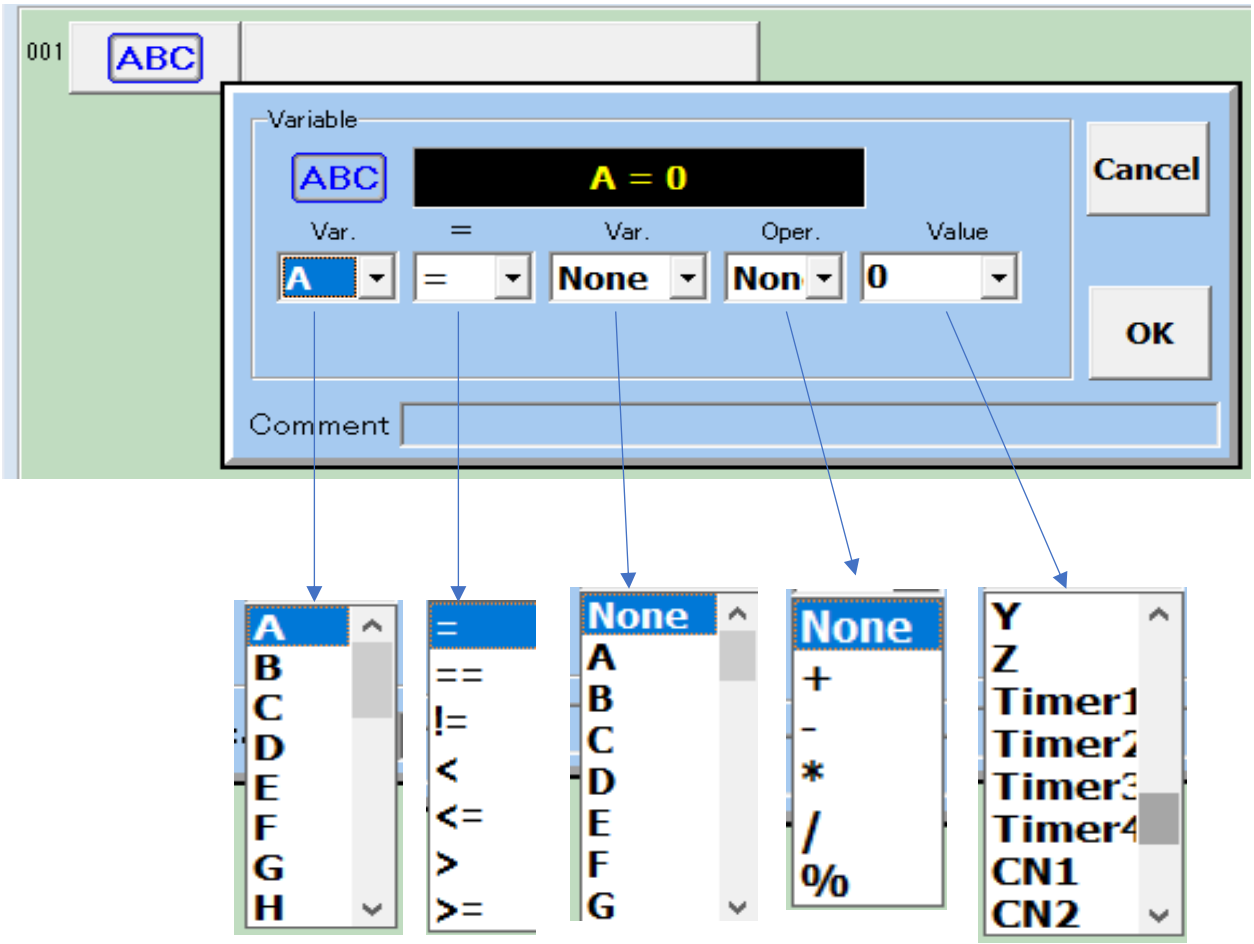
The timer start button serves to start a clock, so that your program can know how long something has been going on. For example, you can program to move in a certain direction for one second, but at the same time check the eye sensors and make sure the robot does not bump into obstacles during second.
There are 4 timers which you can set to measure the time of different actions.
A timer works like pressing the start button on a stop watch. It will continue running until your program resets the timer.



Program example using timers – try this with your robot as an exercise!
This program sets the motors to move forward while the "while" loop is running. The "while" loop runs as long as Timer 1 has not yet run for 1 second. Hence, the motor moves forward for 1 second. So, this "while" loop does the same thing as if you simply inserted a 1 second wait time. However, inside the "while" loop, you can tell the robot to do other things. In this case, we use Timer 2 in order to blink the red LED by turning it on and off every 0.1 second. Does it blink 5 times? You could add even more command buttons inside the "while" loop for the robot to do while driving forwards for 1 second.
The time check in program lines 004 and 005 are explained in chapter 2.10 "Time check" in more detail.

2-6 Variables

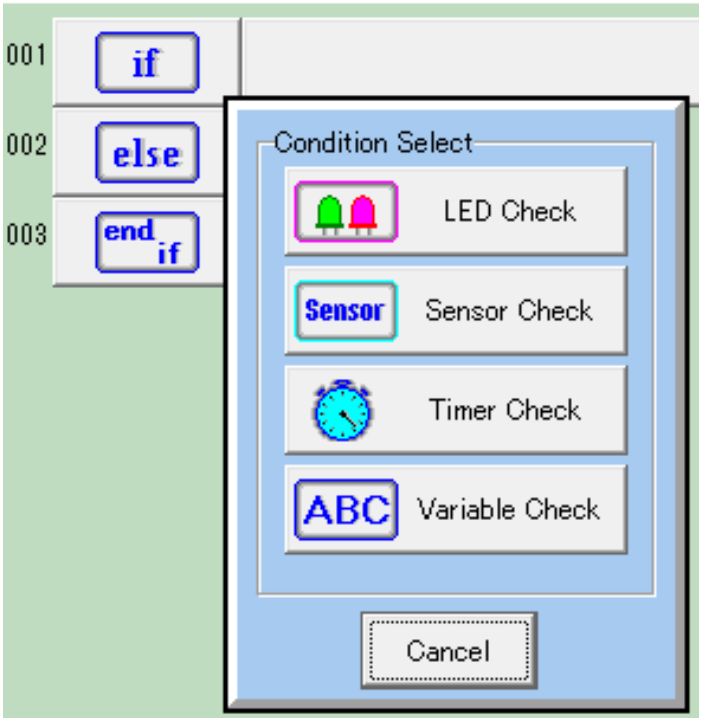
You can use 26 variables in C-Style, named A-Z. These variables store numbers, and can be used for calculations with + (plus), - (minus), * (times), / (divided) and % (remainder). The variables can be used to store timer values and sensor readings (CN1 ~ CN10) for use in your program.



Examples for setting variables:

A = 0	Variable A is set to 0
A = 100	Variable A is set to 100
B = A	B is set to the value of A
A = Timer1	A is set to the current value of Timer 1. (Timer 1 will keep counting, but A will not change.)
A = A + 1	Add 1 to the value of A
A = A * 2	Doubles the value of A
A = A / 2	Sets A to half of its own value
A = B % 2	Sets A to the remainder of B divided by 2. (A will be 0 if B is an even number, and 1 if B is an odd number.)
A = B – CN2	Sets A to B minus the current reading of the sensor connected to CN2. (Note that CN2 has a value between 0 and 4095, it is not a percentage.)

2-7 Condition check – "if", "else if" and "else"



You need to use an "if" command, for example, when you want to program your robot to move backwards when the eye sensor detects an obstacle.

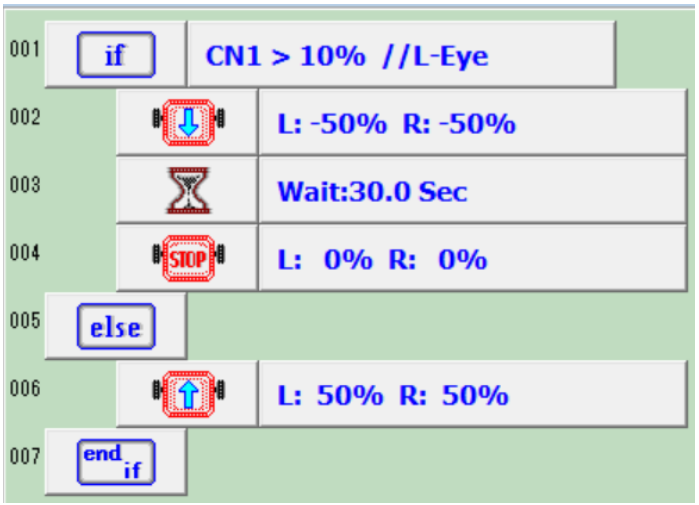
An "if" command has several parts and ends with an "end if". You can add additional command blocks between those parts.

"If" checks a condition, such as the strength of a signal from sensor. Only if the condition is true (for example, the eye sensor detects a strong signal), the commands inside the if statement are executed (for example, you could make the motors move backwards for 0.3 seconds). If the condition is not true (for example, no object is detected), the code inside is skipped over until the corresponding "else if", "else" or "end if"

Every time you use an "if" or "else if" command, you need to select the condition to check. You can check an LED, a sensor value, a timer value or the value of a variable.

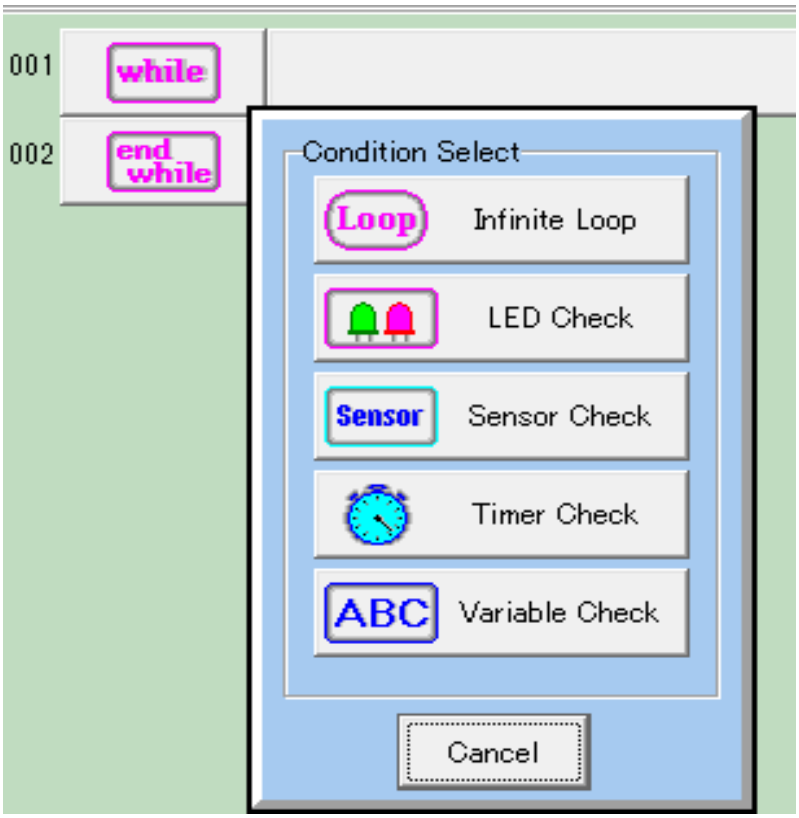
"else if" works exactly like "if", but it is only executed if the initial "if" condition was false. (In our example, we don't need an "else if".)
"else" marks a block of commands that is only executed if the "if" condition was not true and any "else if" checks were also not true. (In our example, you could use the "else" to move the robot forward when no obstacle is in sight.)
"end if" is where the program will continue after the right command block inside the "if" statement has been executed. The "end if" block appears automatically when you place an "if" command.

Once you have set the condition, you need to add the command blocks that you want the robot to carry out if the condition is true. To do so, select the desired command button from the program button list, then click on the position where you would like to insert the command. For example, click on he "else" button in program line 002 to insert command above it (for example, a motor button to go backwards, and then a wait time command set to 0.3 seconds).



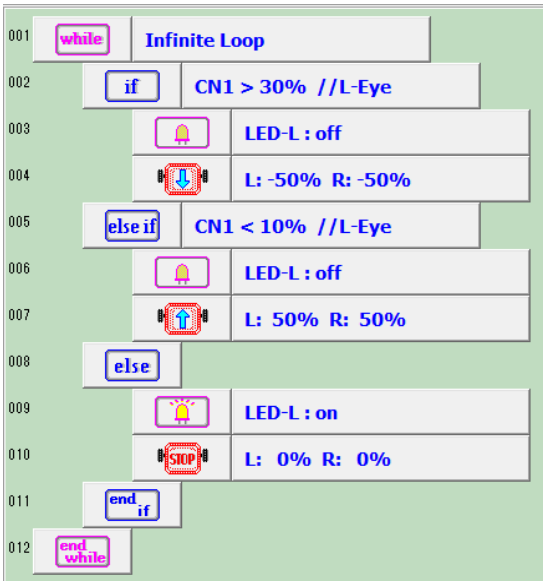
Inserting command buttons is explained in more detail in chapter 1.

2-8 Repetitive action based on a condition – "while"



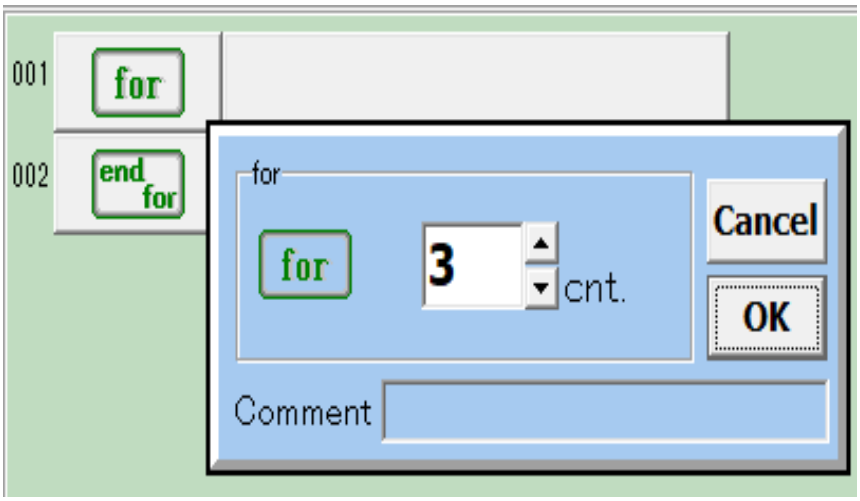
"while" is like the "if" command, carrying out code based on a condition, but "while" executes its command block not only once. "While" repeats the commands between "while" and "end while" until the condition is no longer true. If the condition is set to "Infinite loop", the enclosed block of commands will be repeated until the robot's power is turned off. (Often, a robot's main program is enclosed in a large infinite while loop, because programmers don't want a robot to get to the end of its program before its mission is complete. For example, a soccer robot should not get to the end of its program in the middle of a soccer game.)

Program example with "while", "if", "else if" and "else"



The code between "while" (001) and "end while" (012) are repeated without condition, meaning forever until the power is turned off.
002-004: If CN1 (the L-Eye sensor) reading is higher than 30% (indicating an object in front),
Turn off LED-L
Drive backwards
005-007: if CN1 (L-Eye sensor) reading is less than 10%,
Turn off LED-L
Drive forward
008-010: If both conditions above are false (i.e. CN1 is between 10% - 30%),
Turn on LED-L
Stop the motors

2-9 Repeat action a set number of times – "for"

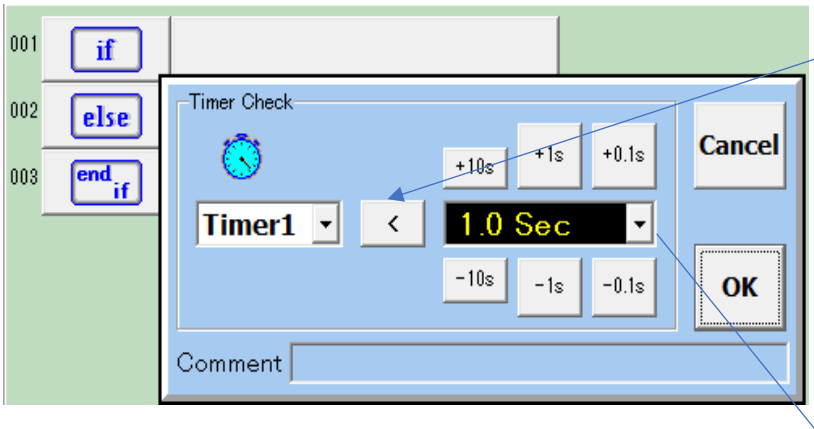


Commands between "for" and "end for" get repeated a specific number of times. Just like you did with the "if" block, insert the commands you want to have executed repeatedly by clicking the first command on the command button list and then clicking on the "end for" block (line 002 in the picture).

"Break" is a command that can terminate a loop and skip to "end for" and to the following commands outside of the "for" loop, even if the command block in the "for" loop has not yet executed as many times as required. "Break" works the same way in both "for" and "while" loops. (The "break" command should always be located inside an "if" block that is inside a "for" or "while" loop.)

"Continue" is only available in Advanced Mode. It makes the program skip back to the "for" of "while" block, thus skipping the remaining commands of the current iteration.

2-10 Time check

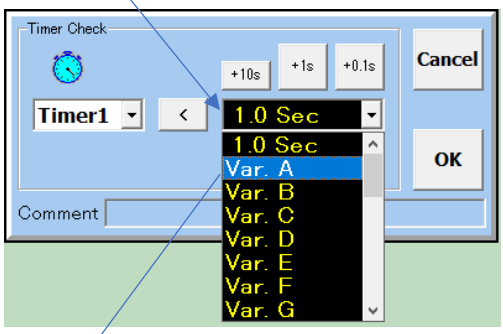


Click here to switch between "<" (less than) and ">" (larger than)

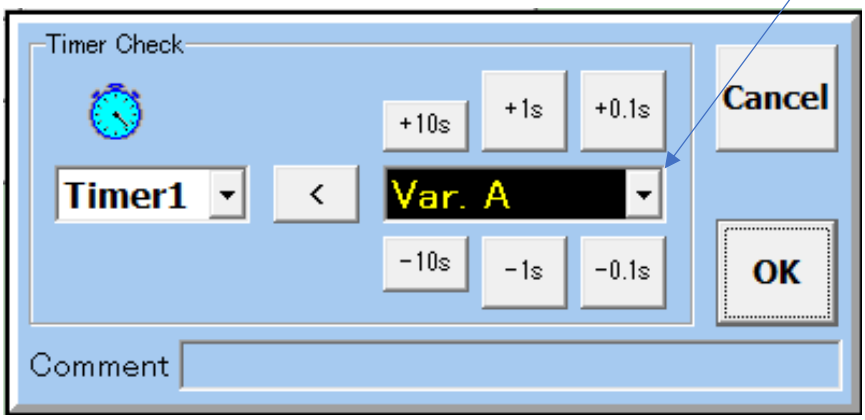
You can use any of the 4 timers and use it for the conditional check in an "if" or "while" statement. Instead of comparing a timer to a fixed amount of time, you can also compare the timer to any of the variables (A-Z).

Examples

Timer1 < 0.1 Sec	Is timer 1 less than 0.1 seconds?
Timer2 > 1.5 Sec	Has timer 2 run longer than 1.5 seconds?
Timer1 > Var. A	Is timer 1 greater than variable A?
Timer2 < Var. A	Is timer 2 smaller than variable A?

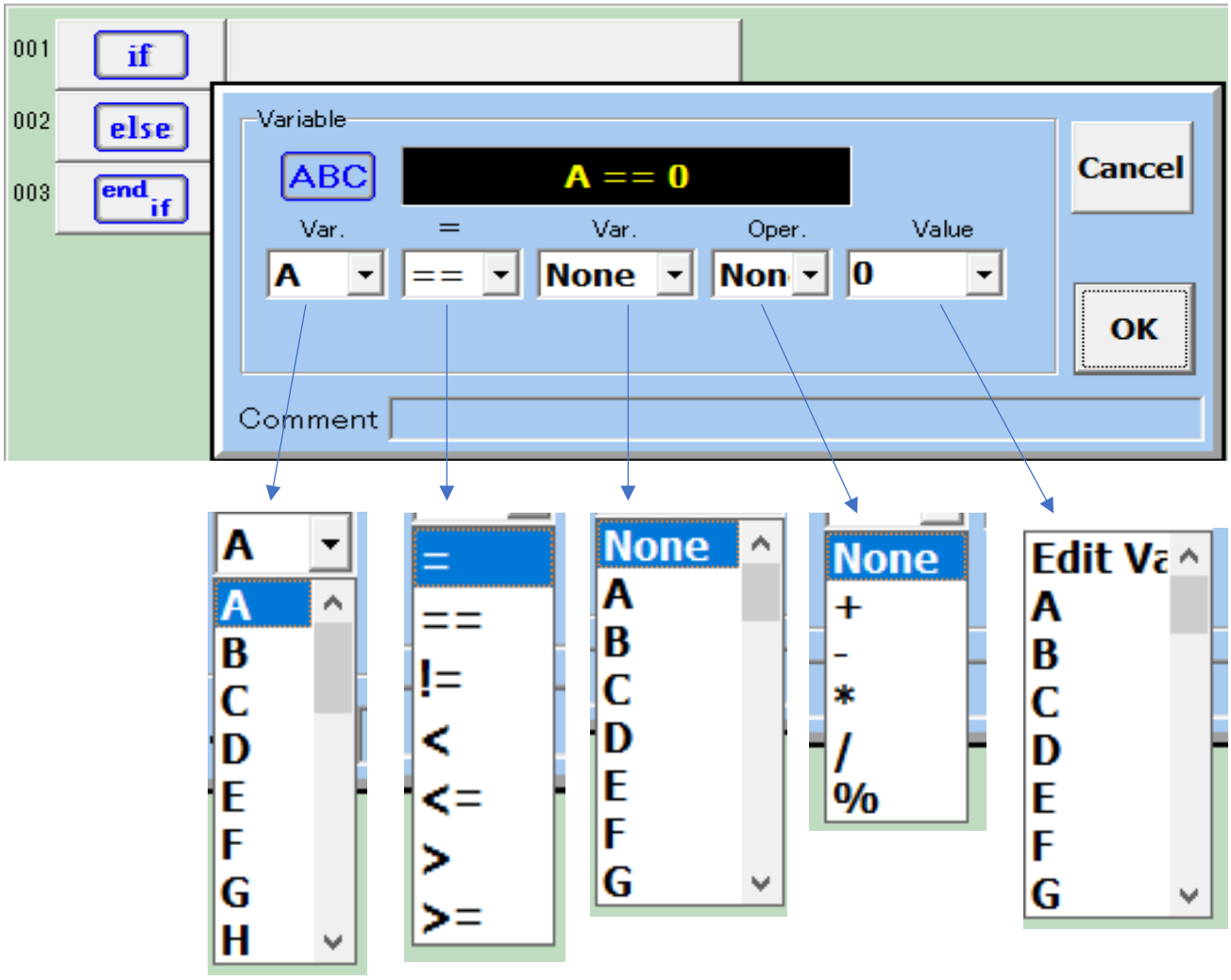


Attention when comparing a timer to a variable:
Variables get interpreted as milliseconds. For example, A = 1500 corresponds to 1.5 seconds on the timer.



2-11 Variable check

C-Style can use 26 variables, named A-Z. The value of each of these variables can be checked by comparison against other variables, timers, sensor input or a combination of those.



Examples of variable checks

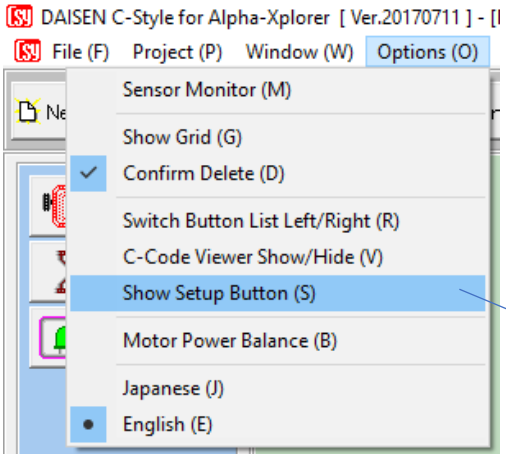
	QUESTION	YES - conditional code executed if	NO - Conditional code skipped over if
A == 0	Is the value of A equal to zero?	A has value 0	A has any value other than 0
A < 5	Is A smaller than 5?	Value of A is smaller than 5	A is 5 or larger
A >= B	Is A greater than or equal to B?	Value of A is not smaller the value of B	B is greater than A
A < CN1	Is A smaller than the sensor reading connected to CN1?	The sensor connected to CN1 gives a greater reading than the value of A	The CN1 sensor reading is smaller than (or equal to) A
A < B-CN2	Is A smaller than the value of B minus the sensor reading on CN2?	The value of B is greater than the value of A plus the reading on CN2	B is smaller than (or equal to) A + CN2
A == B % 2	Is A equal to the remainder of B divided by 2 (0 if B is even, 1 if B is odd)	B is an even number if A was set to 0. B is an odd number if A was set to 1	If the value of A is 2 or more, this test will always be false
A > CN3	Is A larger than the sensor reading on CN3?	The CN3 sensor reading is smaller than A	CN3 is larger than (or equal to) A
A < Timer3	Is A smaller than the current count on Timer 3?	Timer 3 has exceeded the value of A	Timer 3 has not yet reached the value of A

Note:
Variables can be assigned whole number values between -2 147 483 648 and +2 147 483 647 (32 bit integers)
The double equal sign "==" is a common operator in C and other programming languages. It checks whether two numbers are equal. "!=" is the opposite operator, it does the same test but turns true if the numbers are not equal.
Analog sensor readings (CN1 – CN10) are integers between 0-4095, because the Alpha-Xplorer's microcontroller (Cortex M3 STM32F102 ARM) has 12-bit AD-converters.

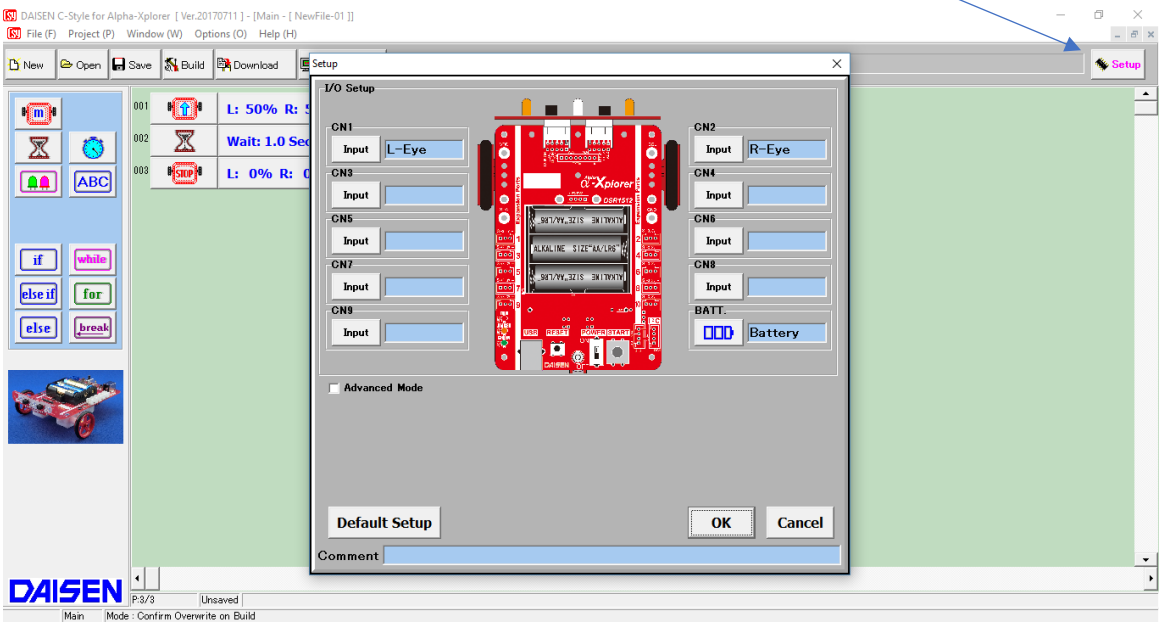
3. Setting up Input, Output an Extended Functionality

Alpha-Xplorer has 10 connections for input and output, named CN1 to CN10.
Initially, CN1 and CN2 are connected to the short distance infrared sensors, CN3-CN9 are available and are set up to measure input from additional analog sensors. CN10 measures the voltage of the battery. These input/output settings can be changed. In particular, CN5 – CN9 can be set up to control servo motors.
In addition, in “Advanced Mode”, you can access the “I2C Device” settings as shown in chapter 3-3 to 3-8. I2C is a digital communication protocol that can control many input and output devices connected to the same connectors on the Alpha-Xplorer.

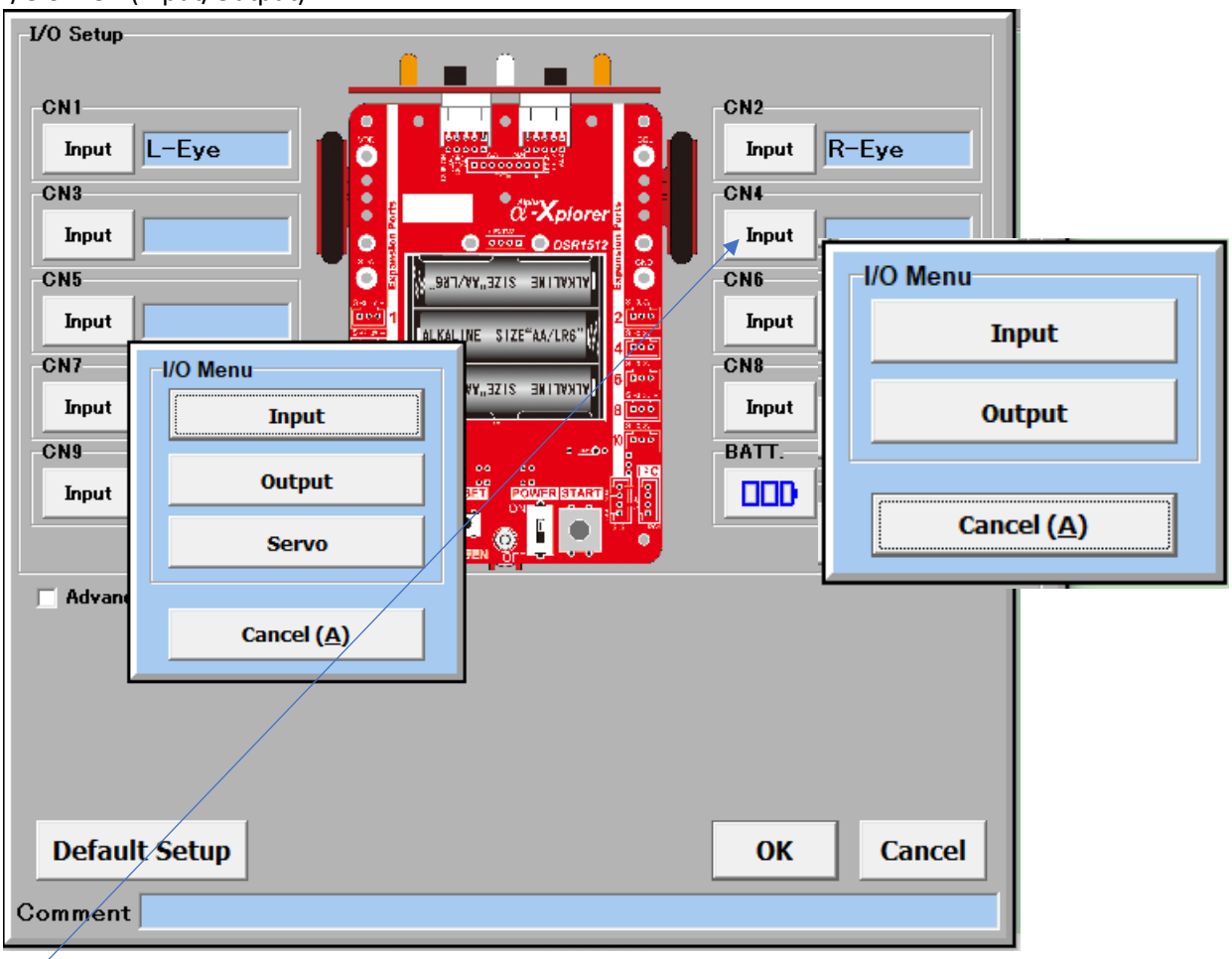
3-1 Setup button



In the “Options (O)” drop-down menu, select “Show Setup Button (S)”.
The “Setup” button appears in the top right corner. Select the “Setup” button to open the “I/O Setup” dialogue.



I/O SETUP (Input/Output)



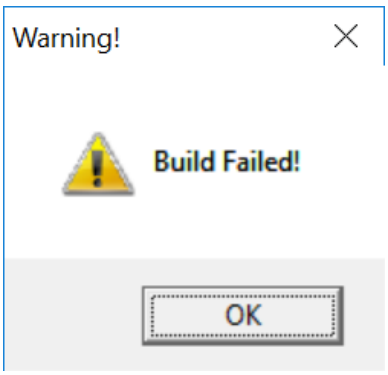
Click on the I/O status button of each sensor (initially labelled “Input”) and a pop-up menu appears where you can set up the connector to listen for input from a sensor, or to provide output to control some action. CN5-CN9 can also be set to “Servo”, which is a special output signal suitable for controlling a servo motor.

If you set any of the connectors to “output”, a command button

will appear on the command button list, so that you can program the output signal for that connector.

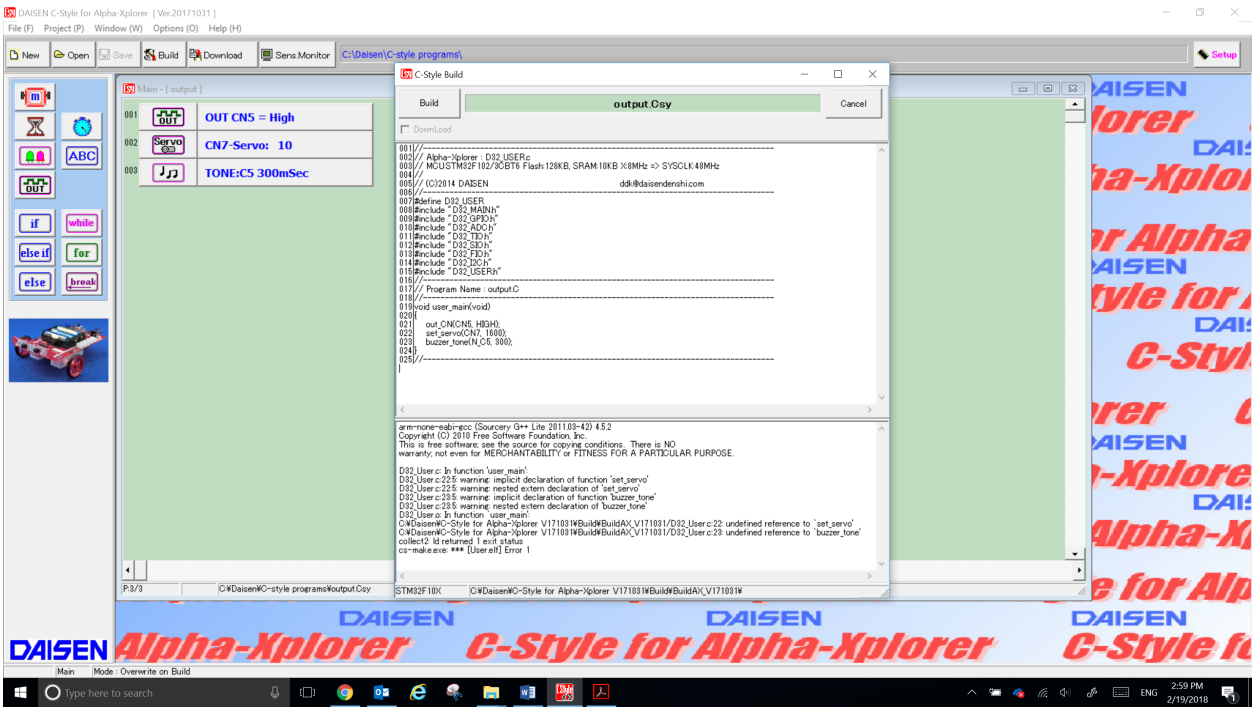
The “Default Setup” button will reset all of the connectors to input, and any output command buttons disappear from the command button list.

Of course, your robot will not know about changes you made to the I/O setup until you have successfully built a program and downloaded it to the robot.

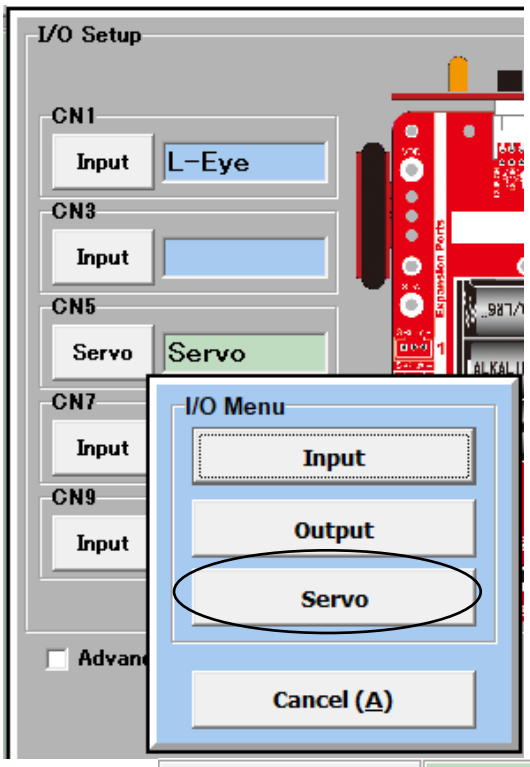


WARNING: If you change the I/O setup after writing your program, the command buttons will disappear from the command button list, but not from your program. This will result in an error and you cannot build your program. The C-compiler will try to send an output signal to a connector that is set to measure an incoming signal. The build process will fail with an error message and will not generate code that you can download to your robot.

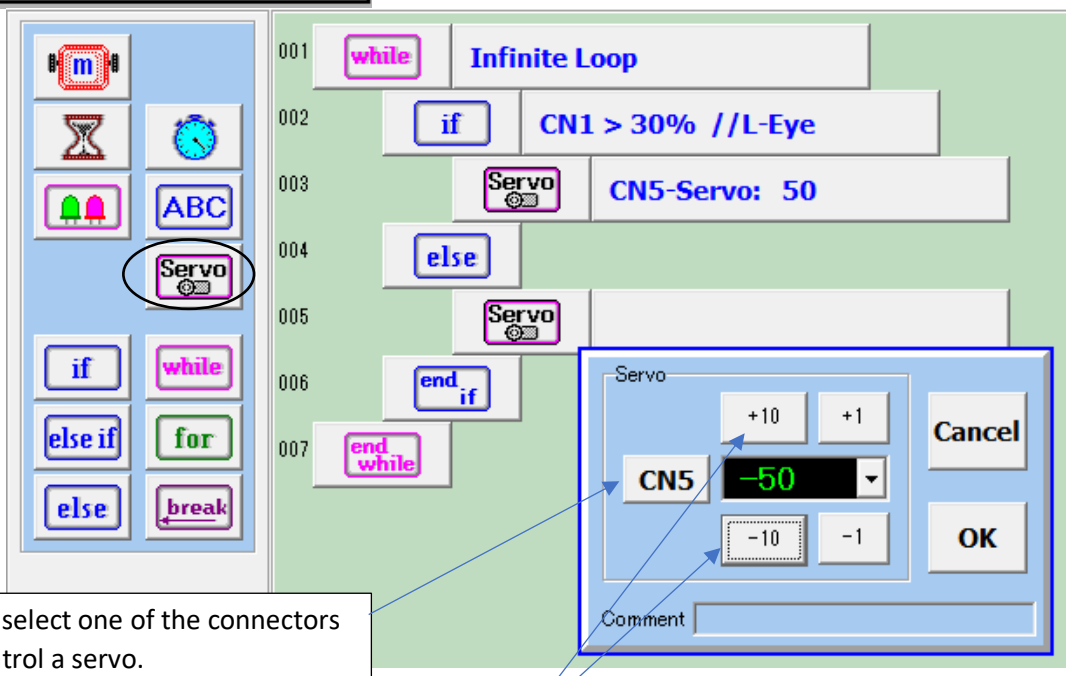
When you write a program where you change the I/O setup in C-Style, we recommend that you write down the I/O setup used for that program. Otherwise, you may get confused when you want to improve your C-Style program a few days later, but don't remember the I/O setup you need to use. (When this happens to you repeatedly, it is probably time for you to move on from C-Style to programming your robot in C, as explained in the C-Style manual No. 3, "C-Style C-code manual")



3-2 Using a servo motor



CN5-CN9 can be set up to control servo motors.
Once set up and used in your program, the program needs to be built and downloaded into the robot. Now, you can use the servo motor and confirm its activity in the sensor monitor.
Servo motors need a separate power supply. Out of the 3 pins on each connector, CN5 – CN9, connect only the signal line (S) and the ground line.
The power pin in the middle of the CN connector is +3.3V and cannot be used to power servos.

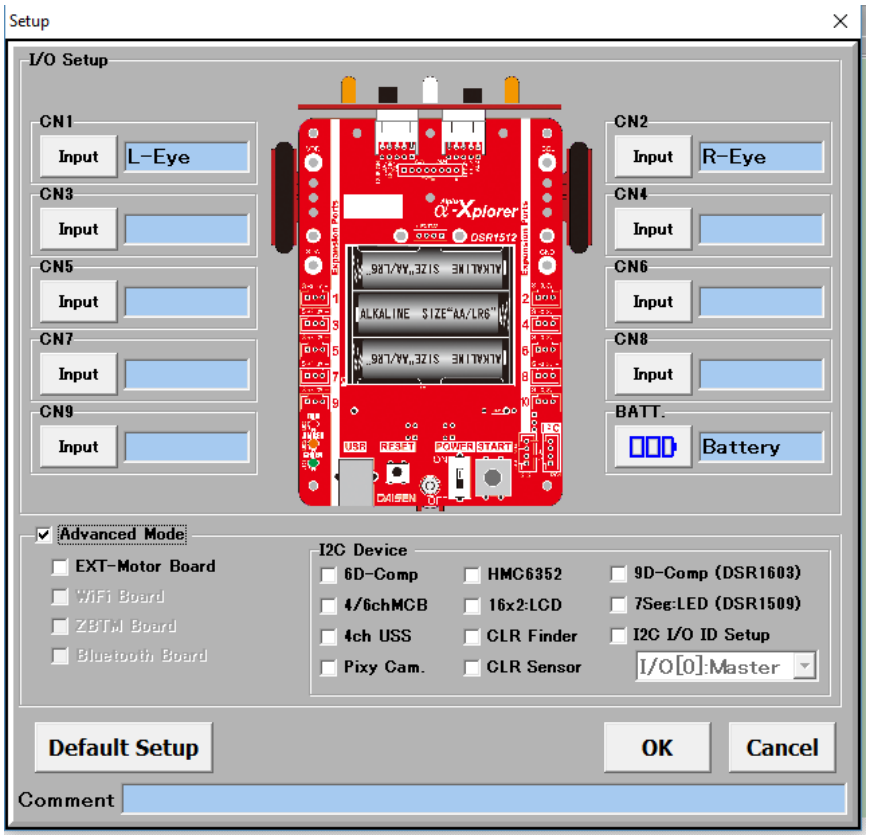


Click here to select one of the connectors set up to control a servo.

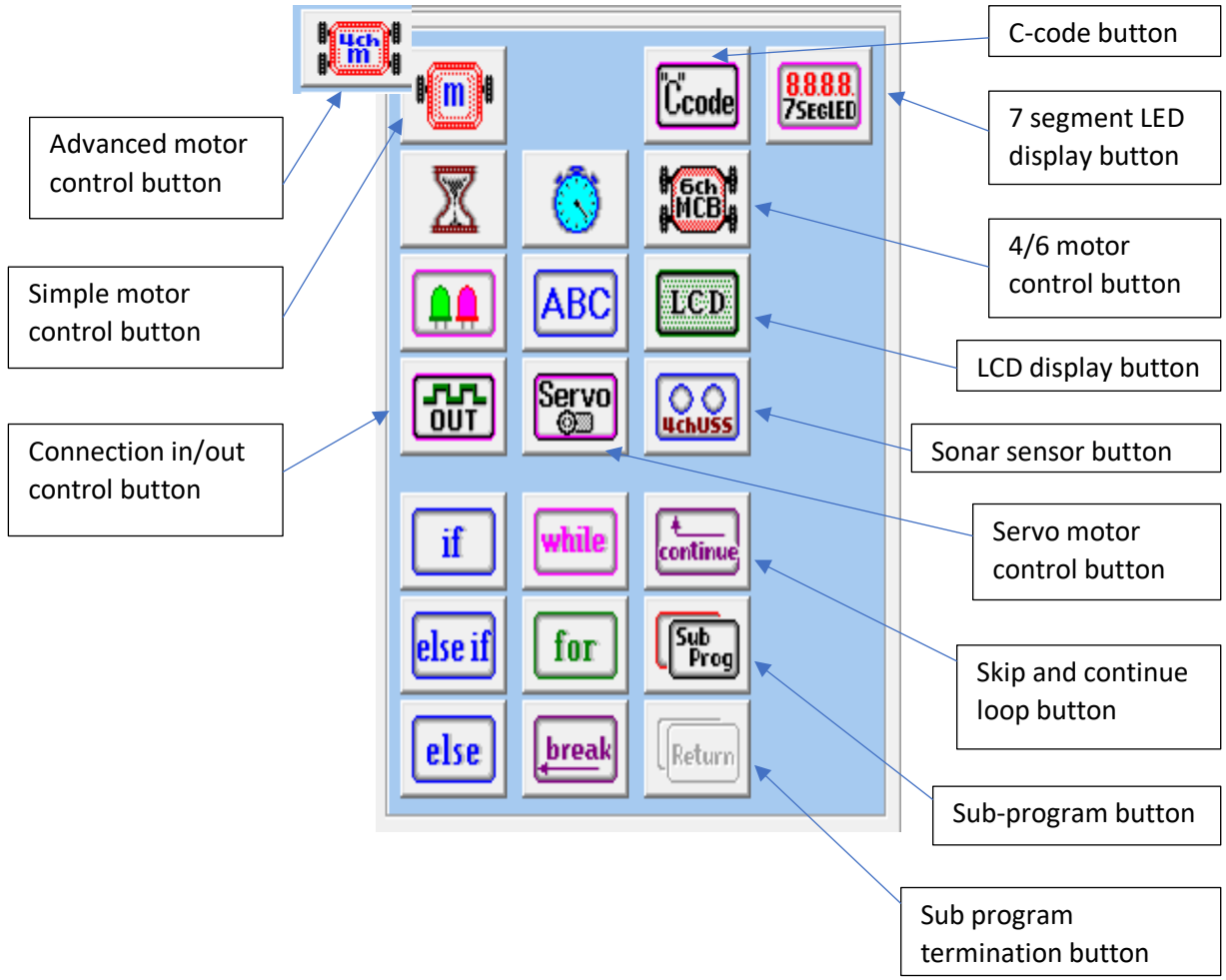
The setting can be between -100 and 100, with 0 being the middle position. Depending on the type and manufacturer, some servos do not move all the way to position 100. Please confirm the range of motion for your servo in the sensor monitor.

3-3 Extended functionality

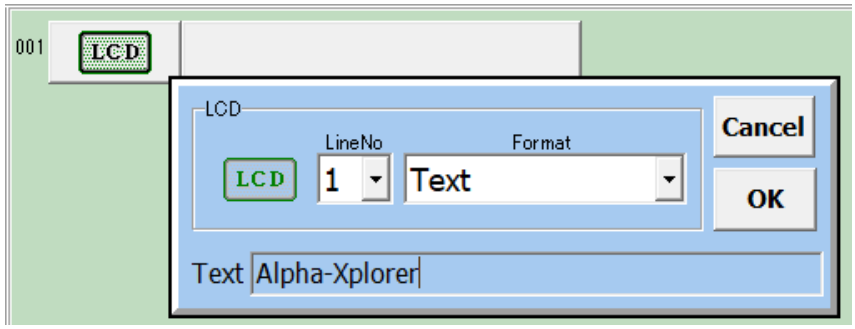
Open the “I/O Setup” window by clicking on the “Setup” button as explained earlier in this chapter.
Underneath the I/O settings to the left is a checkbox “Advanced Mode”.
Click on this checkbox and the extended function checkbox will appear. Check the respective boxes if you want to add to your robot a
6D/9D-Compass (DSR1401/1603)
4/6 channel motor controller board (for example, to build an ominwheel robot)
16x2 LCD display
4ch USS sonar distance sensor
Additional Alpha-Xplorer boards for added functionality via an I2C digital connection

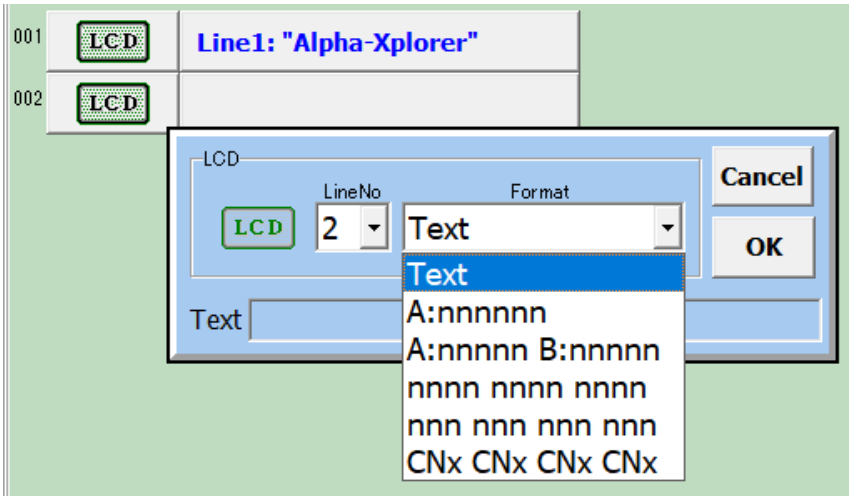


In addition, once you click “OK” to close the setup window, advanced buttons appear in the command button list, including a “C code” button and a “Sub Prog” button. Other advance command buttons appear only if you activate the specific function by checking the box in the “Advanced Mode” panel.



3-4 Using the LCD board (DSR1416)

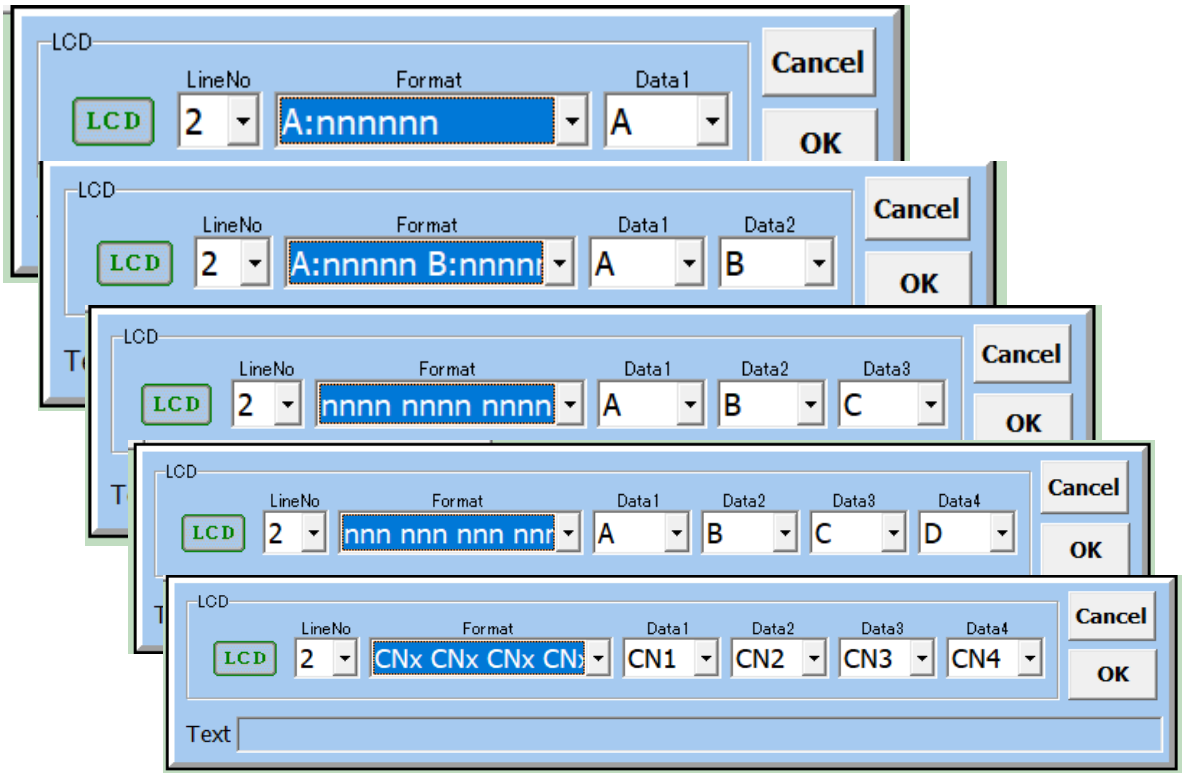




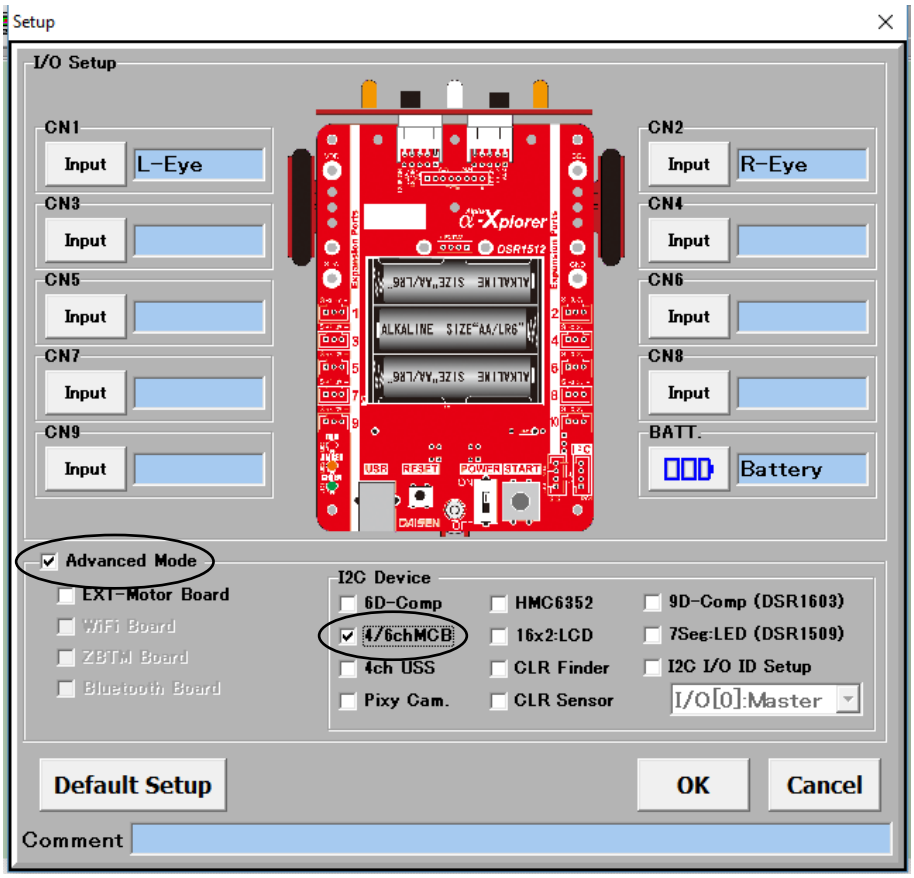
Press the "Setup" button to open the I/O setup window and select "Advanced Mode". In the I2C device list, check the box for "16x2 : LCD", then click "OK". The "LCD" command button will appear in the command button list.

When using the LCD command button, select whether you want to display information in the display's first or second line. Then, in the "Format" box, select what type of information you would like to display. If you want to display text, add the text to display in the "Text" line below. Unfortunately, you cannot display any quotation marks (").

You can also display the content of up to 4 variables and sensor readings.

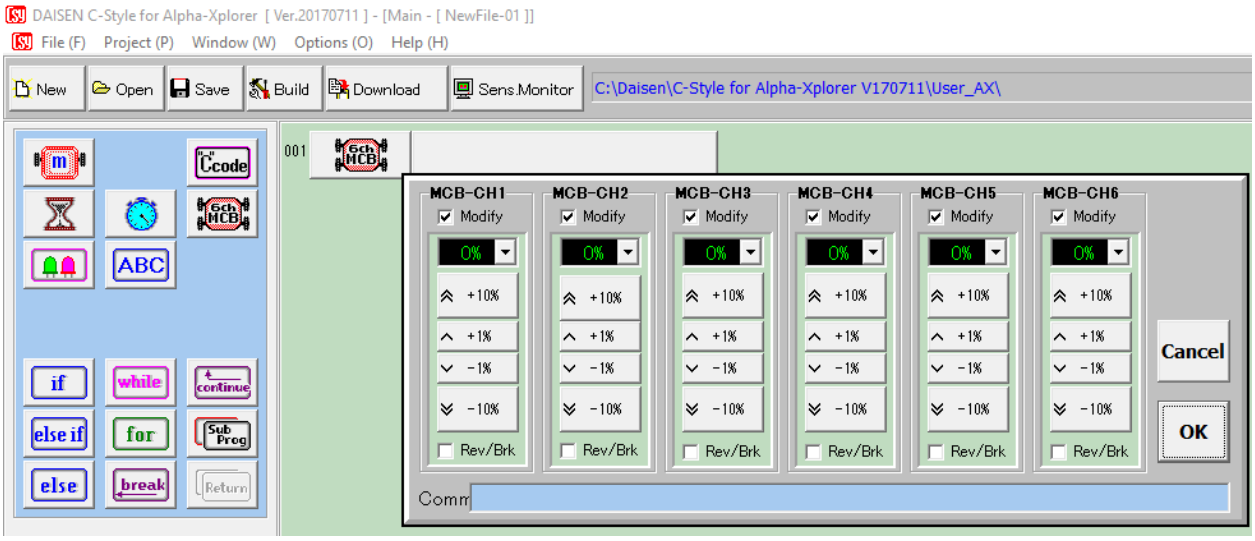


3-5 Using the 4 or 6 channel motor control board



Press the “Setup” button to open the I/O setup window and select “Advanced Mode”. In the I2C device list, check the box for “4ch/6ch MCB”, then click “OK”.

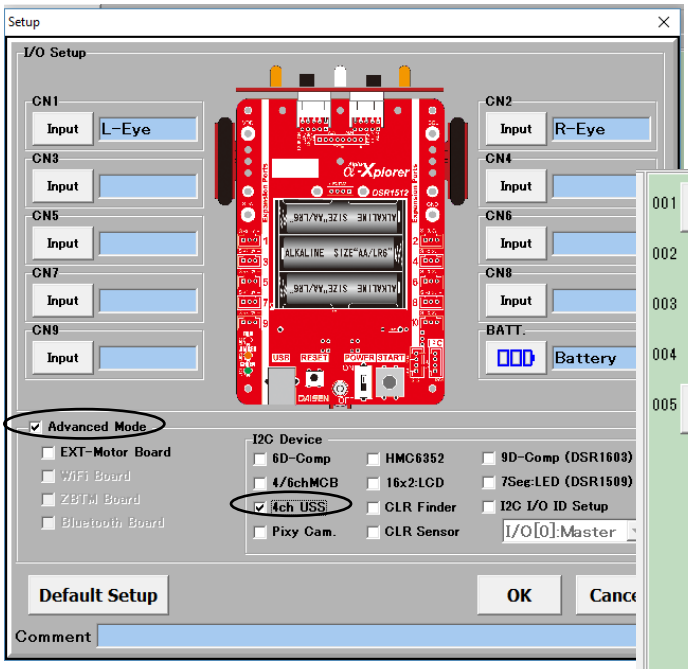
The 4/6 motor control button will appear in the command button list.



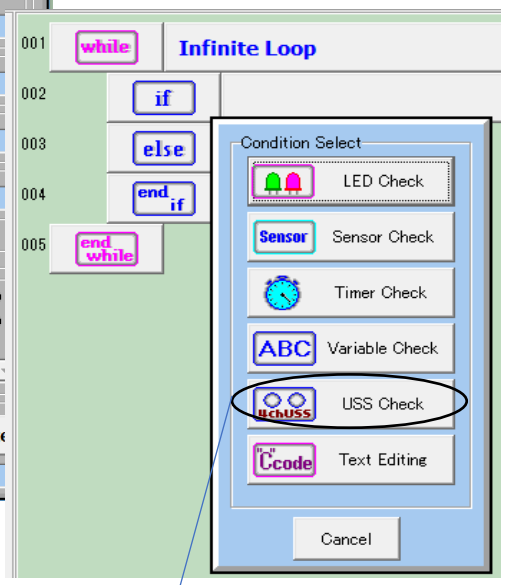
Each motor can be set to a speed between 0% and 100%. To change direction, check the “Rev/Brk” checkbox of the respective motor. Click on the “Modify” checkbox to uncheck any motors you are not using.

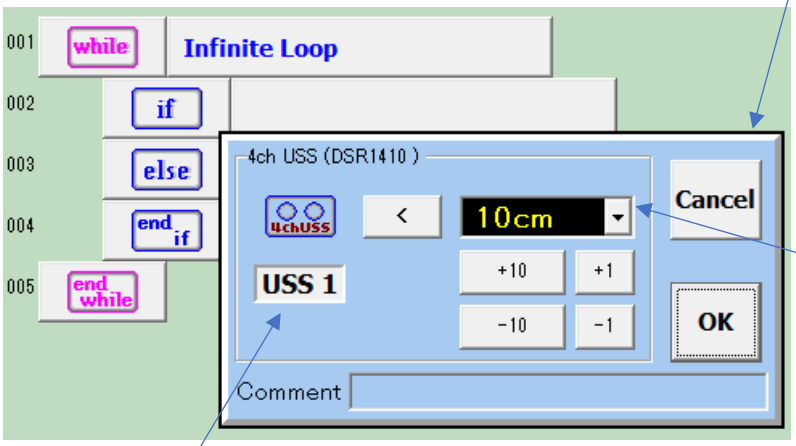
3-6 Using the sonar distance sensor

Press the “Setup” button to open the I/O setup window and select “Advanced Mode”. In the I2C device list, check the box for “4ch USS”, then click “OK”. The multi-motor command button will appear in the command button list.



You can now use the distance measured by the “4ch USS” sonar sensor in your “while” and “if” commands.

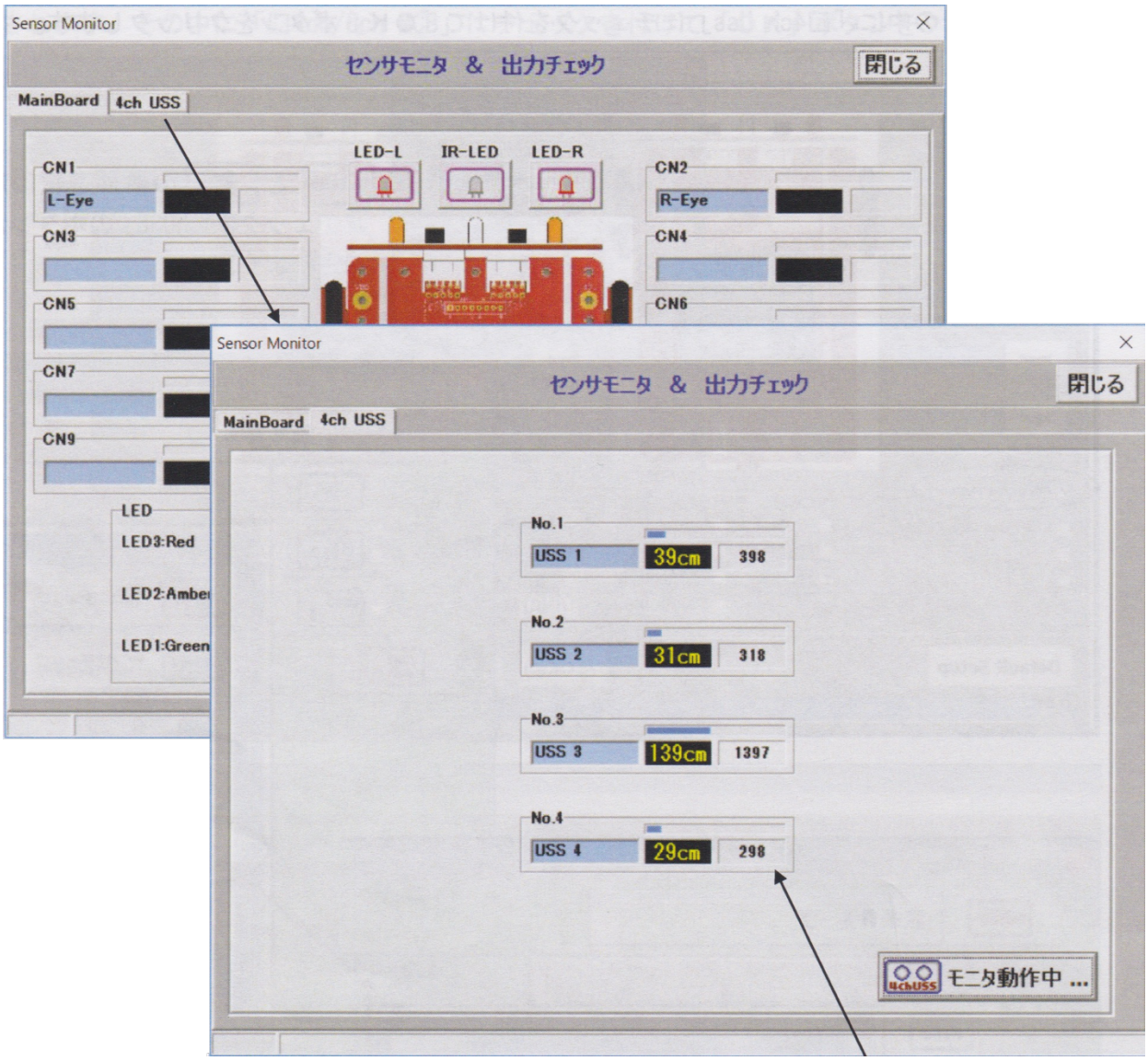




By clicking here, you can select between 4 sonar sensors, USS1 – USS4

Use the pull-down menu to use variables for the condition check, instead of a fixed distance. The value of a variable will be interpreted as distance in millimeters (1/25 inch)

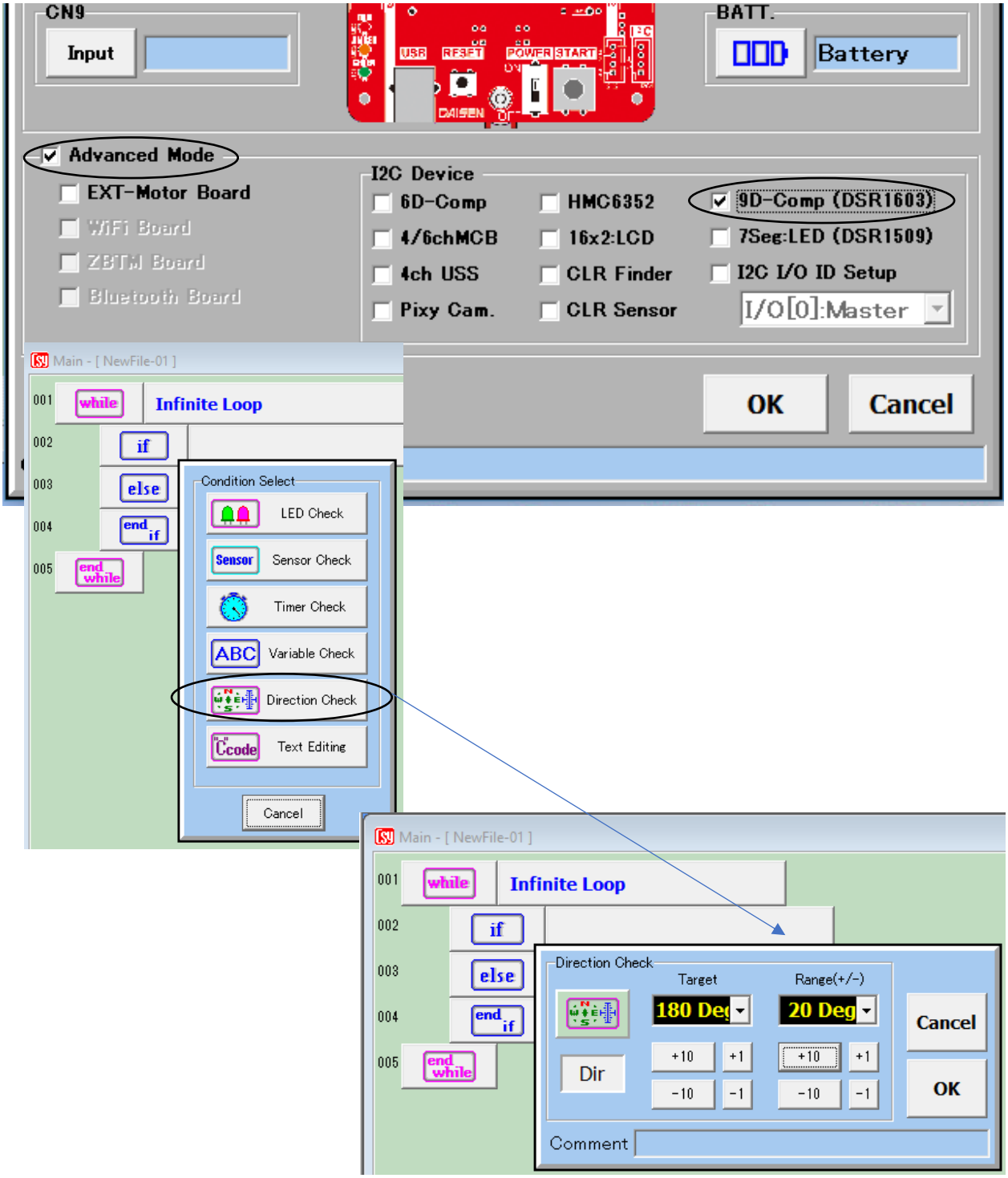
After you set up the “4ch USS” in the I/O setup window, click on the “Sens. Monitor” button. You will see that a new tab appears, called “4ch USS”. Click on that tab to see your sonar sensor readings.



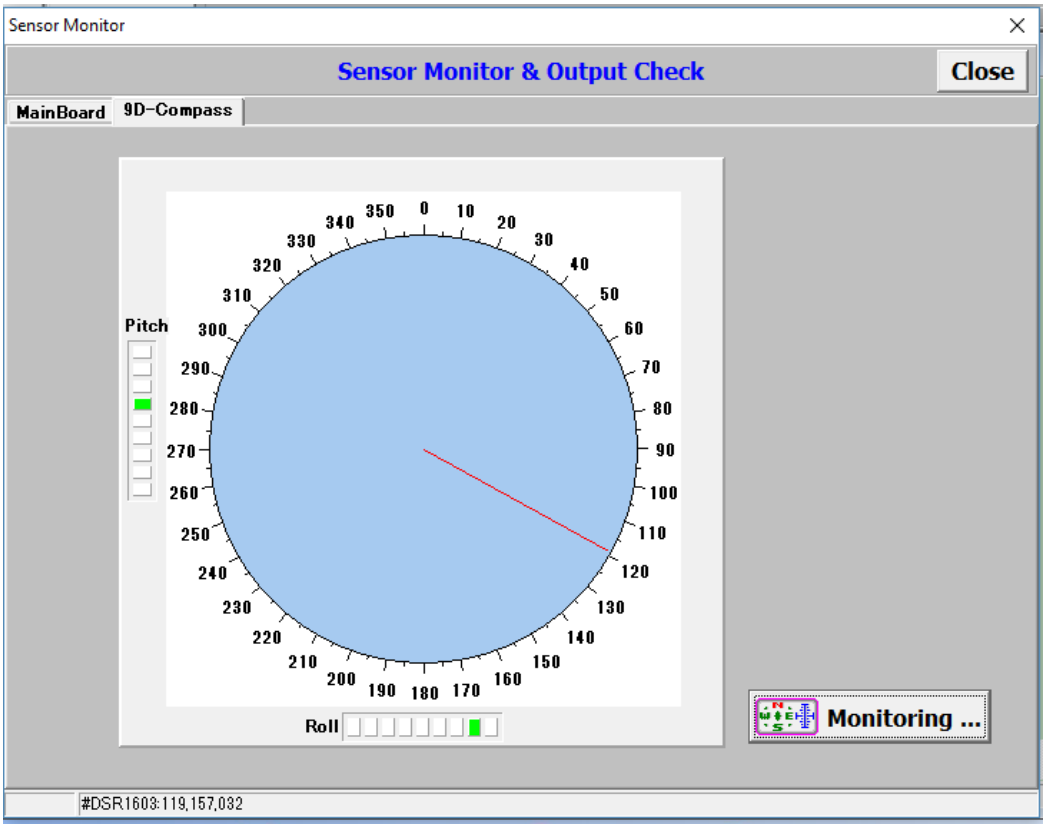
To display the 4ch USS data in the sensor monitor, you need to download an additional software program for your 4ch USS sensor.

3-7 Using a compass sensor (6D/9D-Compass: DSR1401/DSR1603)

To use a multi-function compass sensor, check the appropriate box in the Advanced Mode menu. Now, you can use a direction check as the condition in an “if” or “while” command. Select the “Target” box to your direction of interest, and then the “Range (+/-)” box to the right precision – don’t set the range to narrow at first.



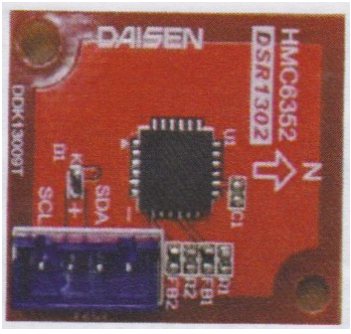
After you set up your compass sensor in the I/O setup window, click on the “Sens. Monitor” button. Click on the compass tab to see the compass sensor readout window. With your robot connected, click on “Start” to see the compass sensor signal.



If the compass sensor monitor fails to display the direction (red line), write any short C-Style program that uses the uses the compass sensor, build it, download it to the robot and run it, to help the robot initializes its connection to the compass sensor. Then, keep the robot connected and go back to the Sensor Monitor. If the compass sensor monitor still does not respond, make sure the 4 contacts of the compass sensor are connected correctly, SDA to SDA, SCL to SCL, plus to plus and ground to minus.

HMC6352

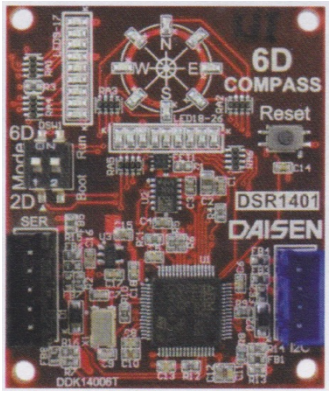
Single Function Digital Compass: DSR1302



Without the function of Pitch/Roll

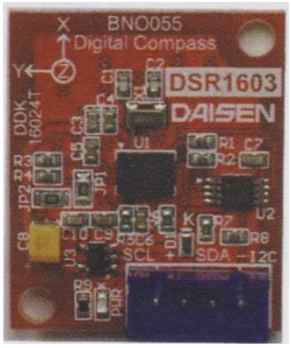
6D-Compass

Multi Digital Compass: DSR1401



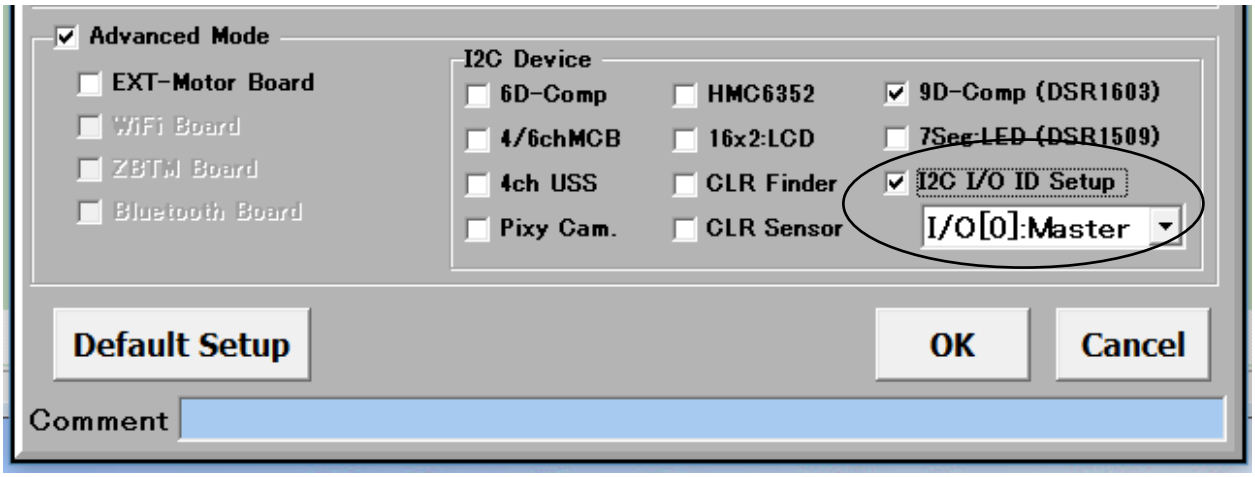
9D-Compass

Multi Digital Compass: DSR1603

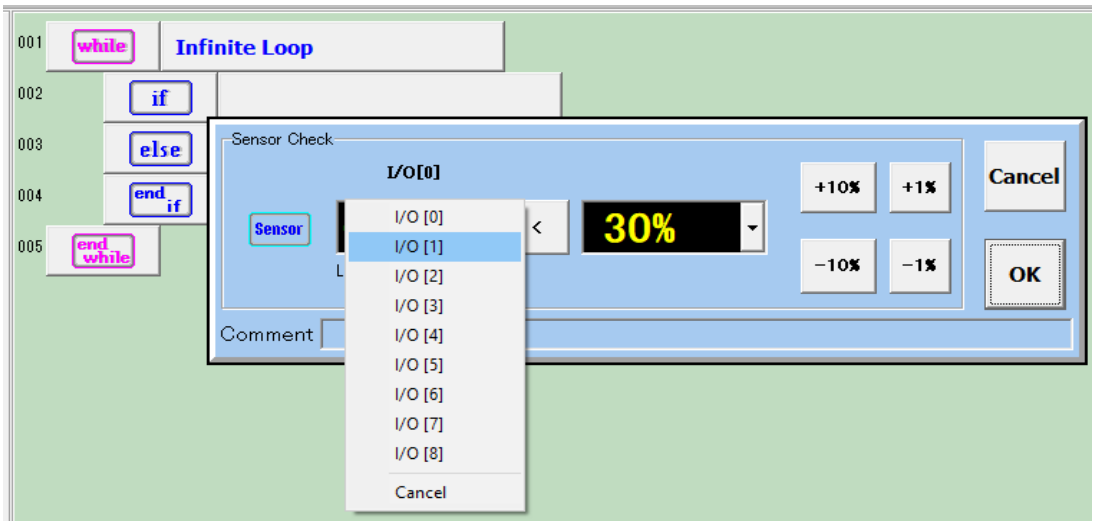


3-8 Connecting several Alpha-Xplorer boards

Check “I2C I/O ID Setup” box if you would like to another Alpha-Xplorer board as an additional I/O sensor board.



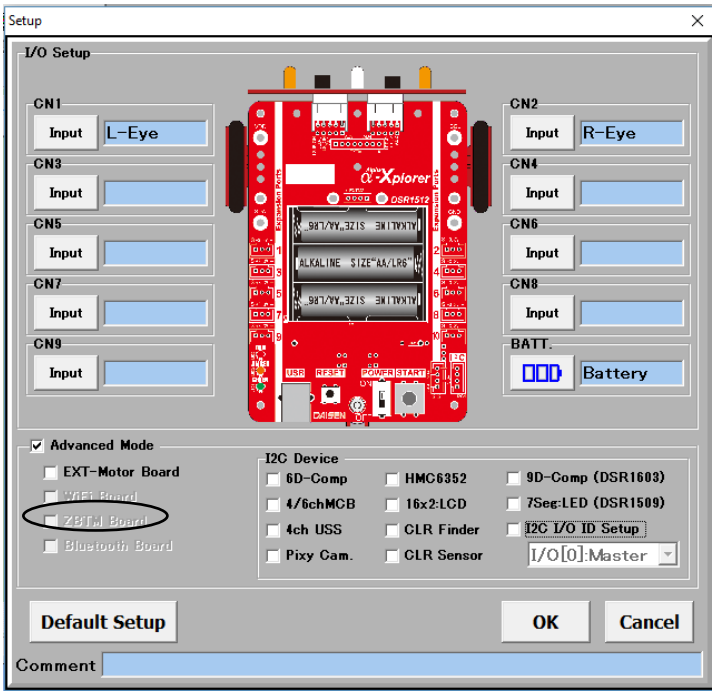
A single Alpha-Xplorer can use 9 analog sensors, connected to CN1-CN9. However, you can connect another Alpha-Xplorer through the “I2C” 4-pin digital connector to add more sensors.
The main Alpha-Xplorer which handles information from all the sensors is called “Master” (I/O[0]). Additional Alpha-Xplorers which send sensor information to the parent are called Sub-I/O (I/O[1] ~ I/O[8]). First, in the setup window, we need to decide for each Alpha-Xplorer whether it is the Master or a Sub-I/O, and if so which number.
As long as the Sub-I/O program is built and downloaded with the correct ID number (1-8), corresponding to the Sub-I/O number set up in the Setup window, it will run by turning on the power (without pushing the start button on the Sub-boards) , and the Master will be able to read the sensor information from the Sub-I/O boards and use it in its program.



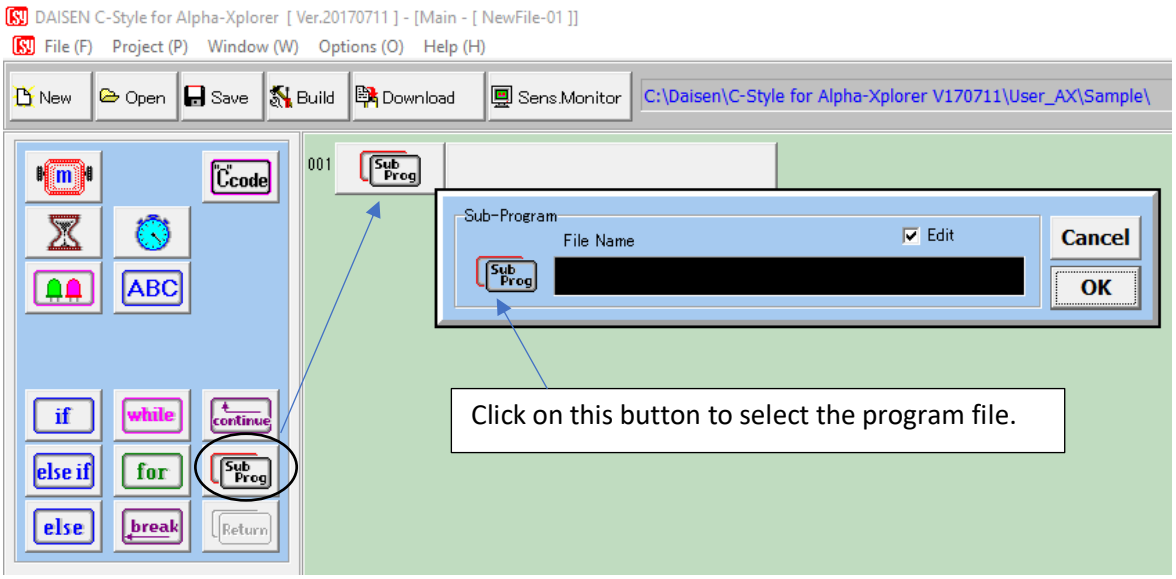
4. Sub-Programs

4-1 Finding the sub-program button

Press the “Setup” button to open the I/O setup window, check the “Advanced Mode” checkbox and click “OK”.



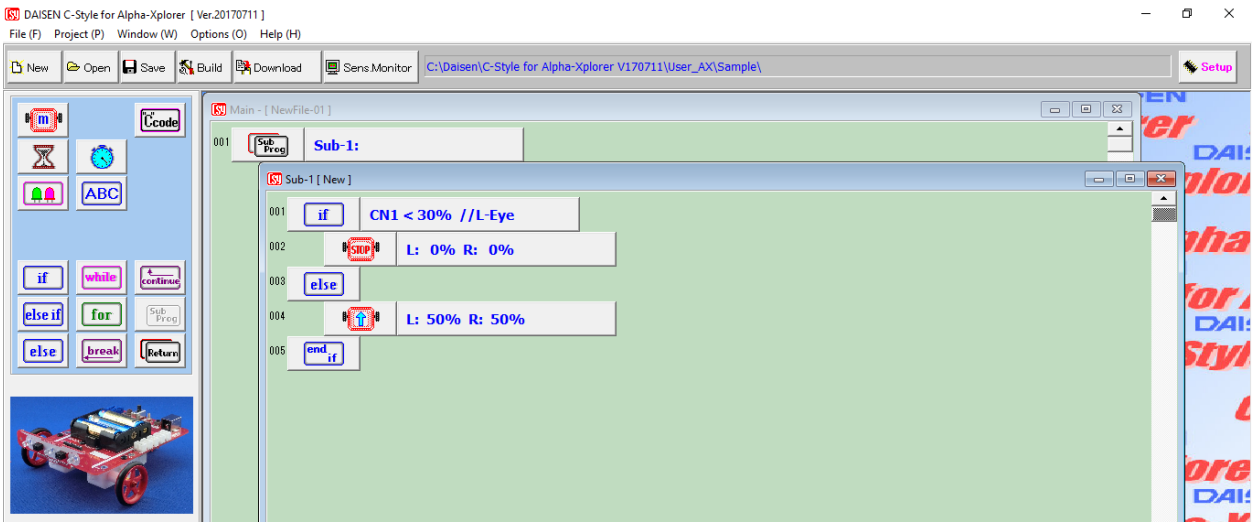
The “SubProg” command button will appear in the command button list and can be used in your program just like the other command buttons.



Sub-programs are very useful if your robot needs to do the same procedure several times. You can program the procedure inside a sub-program, and have only one command button in your main program, the “SubProg” button, to carry out that procedure. (C programmers call this a “function”.)

4-2 Creating sub-programs

You can create up to 30 sub-programs in C-Style. However, C-Style does not allow you to put a “Sub-Prog” button in a sub-program. (If your program gets complicated enough that you need nested subroutines, consider switching to C code, as explained in the next manual.)
To create a new sub-program, position the “Sub Prog” command button in your program and click on “OK”. If you want to us an existing sub-program, click on the “Sub Prog” symbol, and then select the sub-program you need.
Please make sure you store sub-programs in the same folder with the main program.



In the open sub-program window, you can program your sub-program just like the main program, by selecting command buttons from the command button list and placing them in your sub-program.
To save your sub-program, make sure that the sub-program window is active, by clicking your mouse anywhere in the window. The active window’s title bar and frame has a slightly darker blue color than inactive program windows. With the sub-program window active, simply click the “Save” button in the top left area of the C-Style window. Save the sub-program file in the same folder as the main program. The file name you chose when you save the sub-program will appear as comment in the sub-program button in the main program.

MEMO 1

MEMO2

DAISEN

4-9-24, Nipponbashi, Naniwaku,
Osaka, Japan 556-0005
TEL+81-6-6631-5553/FAX+81-6-6631-6886
URL http://www.daisendenshi.com
Email ddk@daisendenshi.com

Translated and edited by
Robomov LLC
252 Nassau Street,
Princeton, NJ, USA, 08542
www.robomov.net
info@robomov.net