

Reference Guide



Screen-free coding with
snap-together electronic parts

*Hi!
I'm Oscar,
the coding
otter.*

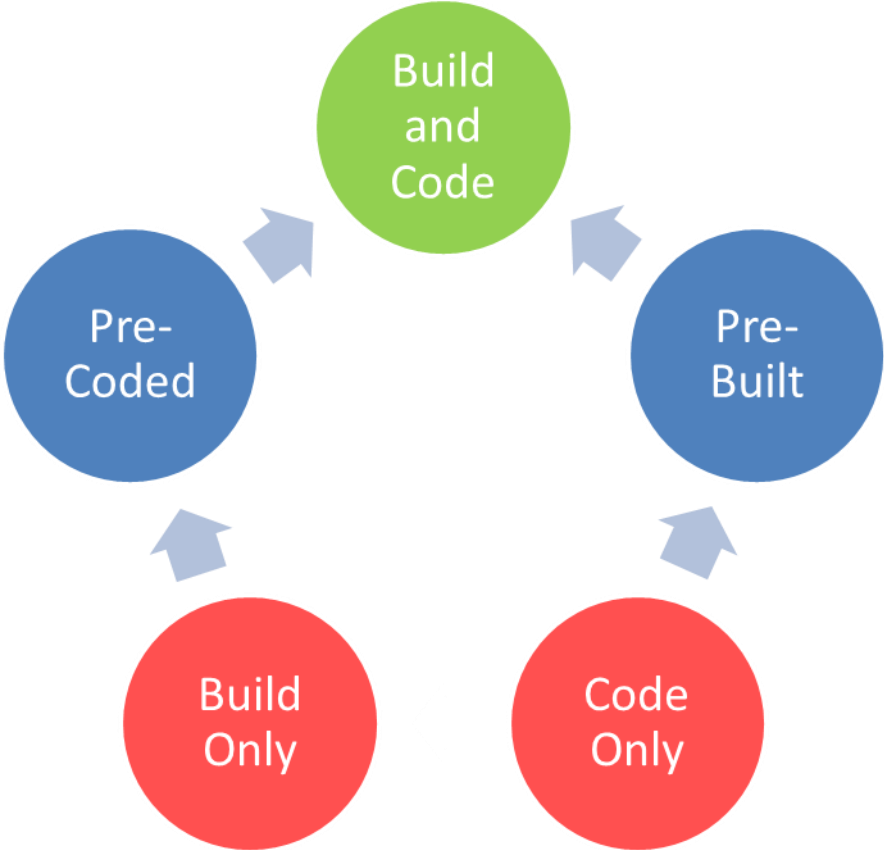


Expert guidance for
parents, teachers and
advanced users.

As we play you
will learn about:

- Actions
- Colours
- Numbers and Sums
- Sounds and Tunes
- Loops and Logic

Coding with Oscar is an early learning coding system from the makers of the award winning KodeKLIX® system and is part of the KodeKLIX® coding and electronics learning suite.



Contents

	Page
Introduction	4
Three Ways to Use the System	5
Getting to Know Your Kit	6-8
Batteries and Powering On/Off	9-10
Standard Coding KLIX	11-16
Advanced Coding KLIX	17-19
Coding Examples	20-28
Coding Do's and Don'ts	29
Troubleshooting	30
Construction Options	31
Brick Construction Techniques	32-35
Electronic Circuit Projects	36
Playing Games	37
Code Scoring	38
Making Your Own Games	39
Game Templates	40-42
Education and Curriculum Linkages	43-
Large Print Format Story Books (and quiz answers)	
Ideas Book One	
Ideas Book Two	
Ideas Book Three	



Introduction

Coding with Oscar is a novel screen-free coding system for children aged 5+.

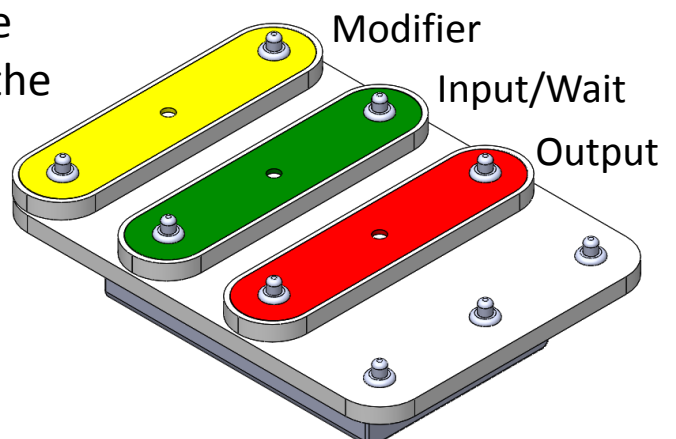
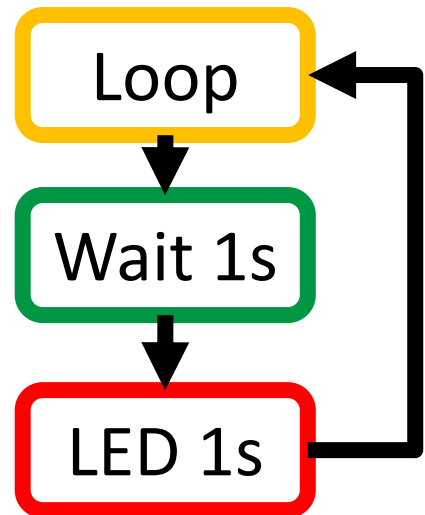
The system requires no apps, tablets or computers – instead the kits use story based learning with tactile snap-together electronic parts to teach real screenless coding skills.

The system uses coding pieces called KLIX. By snapping them to the coding block you can make things happen.

To make your code all you need to do is snap the KLIX in the order you want things to happen. There is no right or wrong way, you just snap them in sequence, from the top to the bottom of your coding block and push the play button to see what the code will do.

There are various kit levels to suit beginners and advance users.

Users can also extend themselves by utilising more construction in their projects or by using the scoring capabilities of the system to play a range of board games.



Three Ways to Use the System

Code Build Play

There are at least three ways to use the system and of course you can combine the approaches for even more depth. The fundamental approach is to encourage play, and through play provide learning outcomes.

Story mode with interactive coding

In this mode, an adult or older child will read the stories and the younger child will follow along with the actions performed by the characters as they introduce the functionality of the coding KLIX. Each short story introduces coding elements and coding theory in a subtle way, challenging the child to experiment in order to learn more. These stories work with all kit versions.

Construction options: art, craft, brick and circuits

The coding block can be decorated to make it personalised, and the ABS plastic allows it to be easily wiped clean. The handheld form of the coding block in Kit 1 can be built into a number of models, constructed from a variety of materials including card and bricks. Kit 2 includes additional parts such as links to allow basic circuits to be constructed.

Interactive board and map games

Built into the coding engine is logic to score the complexity and creativity of the code created by the child. More complex and user interactive code is given a higher score, and this scoring system can be used to play or customise interact table-top games.

To activate the 20 second game count down timer, push the Play button with no KLIX attached to the unit.

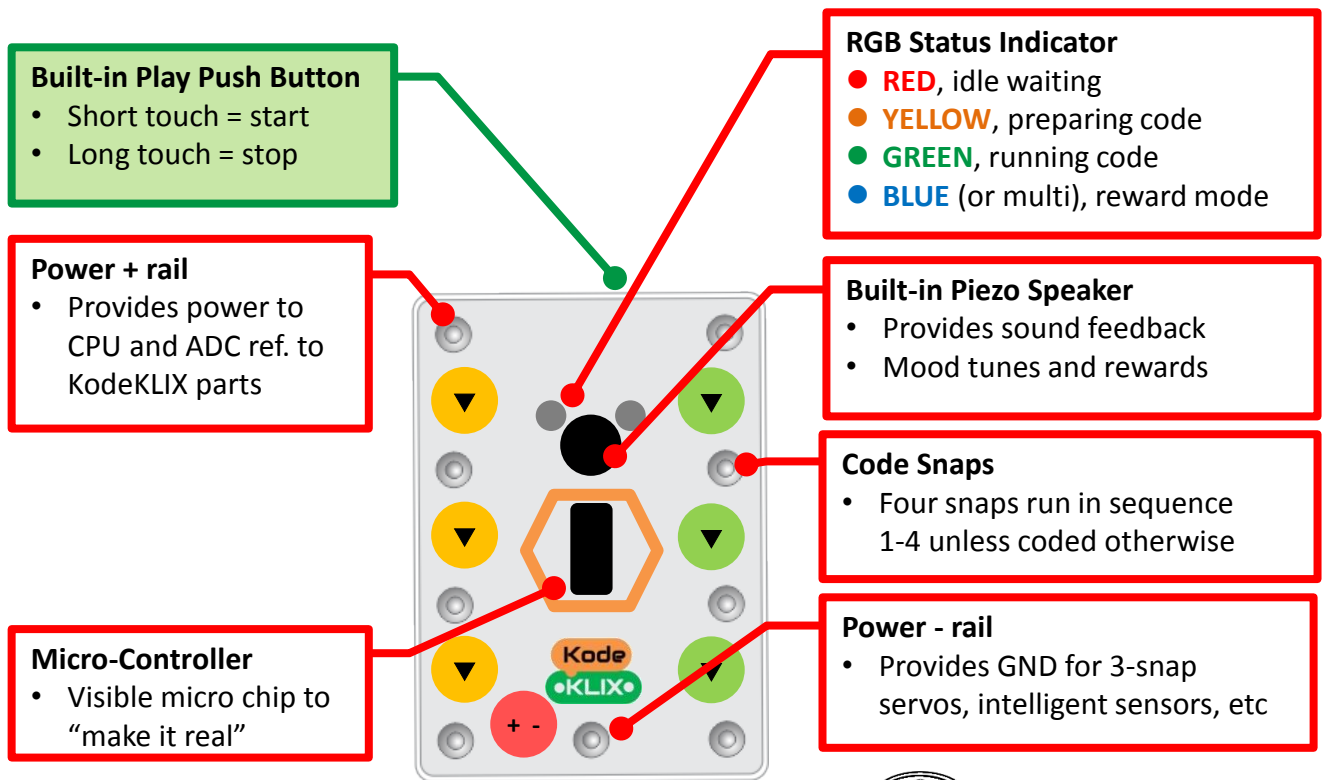


Getting to Know Your Kit

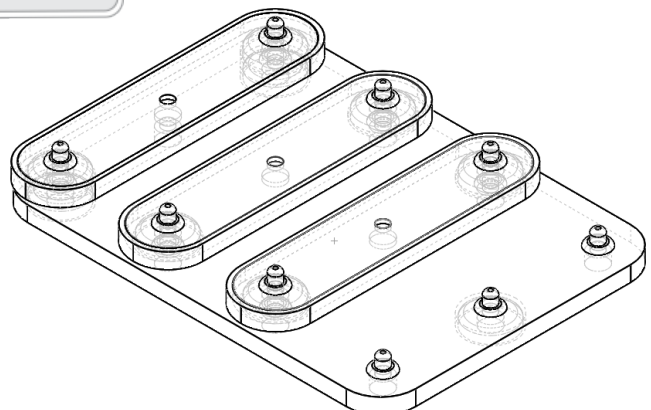
Kit 1: Coding with Oscar Starter

This kit uses a self-contained handheld coding block capable of up to four (4) coding steps. It provides support for all the system coding elements and game engine, but included pieces limit coding complexity to an introductory logic level. The coding block can conveniently be built into interactive models and expanded with additional coding KLIX, as skills develop.

After a short power-on tune, the status indicator will show you what mode the coding block is in. Snap the code KLIX onto the front face, and then press the Play button on the back to run it.



Max. 4 coding steps



How the Coding Works

The following pages provide working examples of how the coding KLIX operate.

Each coding KLIX represents a line of code. Each line is responsible for an action or for modifying an action.

Instead of using words (like languages such as Python, Java or BASIC), or Block pieces on the screen (like in apps such as Scratch or Blockly), *Coding with Oscar* uses physical KLIX to represent these instructions.

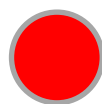
By default, each coding KLIX operates for at least one second – but like real code, this can be modified by the user. You can hear the coding block ticking as code is run. This is the beat of the computer.

The Status LED on the coding block lets you know what its doing. The Status LED has one of four colours, unless its flashing. The first three colours are like traffic lights – Ready, Set, Go!

Begin by assembling your code on the Kode block; selecting and arranging the coding KLIX in the order you want things to be sequenced.

To start the code running, you need to press the Play button (or touch pad).

The code is prepared and then runs.



Idle, waiting – this is when you snap your code together



Preparing code – the code block is reviewing your code



Running code – this means your code is now actioned



Reward mode – code stopped, let's celebrate the achievement

The blue colour shows that code has stopped running, either because you stopped it – or because it finished. More about reward mode later.

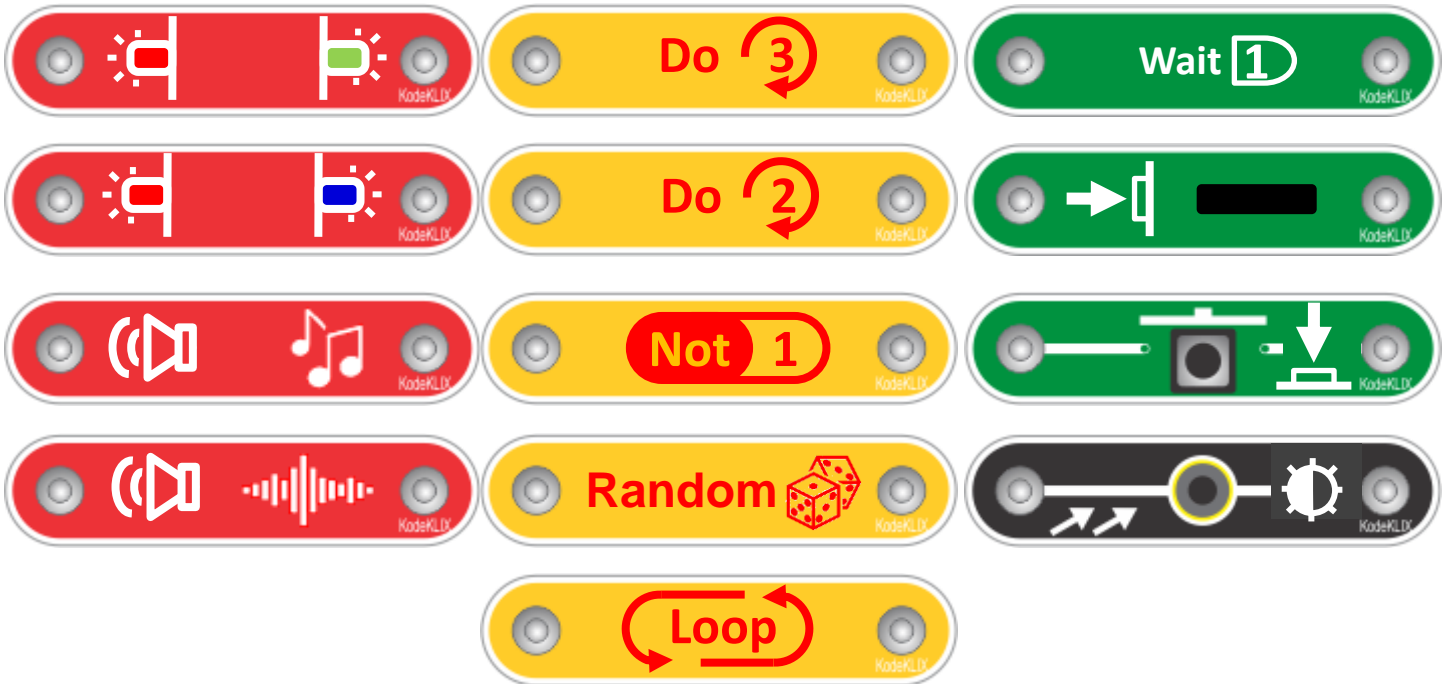
To stop the code running, press and hold the Play button again until the LEDs turn blue.

If you remove a KLIX before the code finishes running, the code block will make an error tone, and no reward will be shown.



Coding KLIX in the System

Coding with Oscar Starter

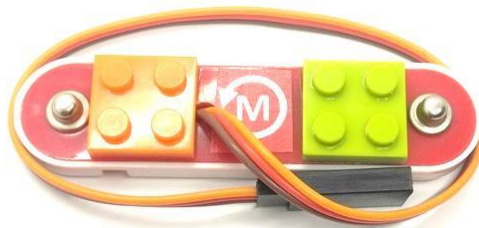


Optional Expansion KLIX



Servo motor options

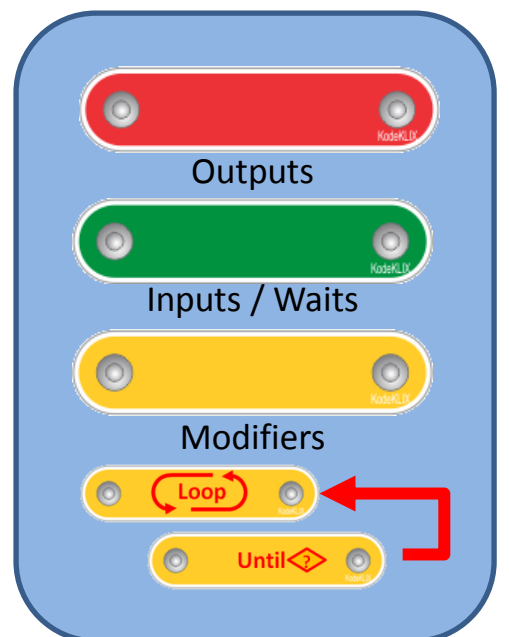
Note: servo and RGB LED KLIX have three snaps and must be orientated correctly to work.



Positional



Rotational



STEM Alignment

Science	Technology	Engineering	Maths
<ul style="list-style-type: none">• Colours• Light / dark• Sound	<ul style="list-style-type: none">• Sequential and repetitive coding• Logic• Function and parameters	<ul style="list-style-type: none">• Inputs and outputs• Actions	<ul style="list-style-type: none">• Counting• Numbers and Sums• Chance and Probability
<ul style="list-style-type: none">• LED KLIX• Light sensor KLIX• Sounds KLIX• Tune KLIX	<ul style="list-style-type: none">• Loop and until conditions• Indexed lookups• Delays & Timing	<ul style="list-style-type: none">• Construction• Design• Automation and control	<ul style="list-style-type: none">• Number KLIXs• Counting blinks• $[1]+[2] = [3]$• Random KLIX

Batteries – Kit 1

Kit 1 is powered by AA-sized user changeable batteries. These represent an affordable and convenient solution. The number and location of batteries varies with the type of kit you have.

On the handheld version of the coding block, you will find the batteries on the rear of the unit. As this unit is aimed at younger kids, a screw is included to lock the battery cover in place; preventing kids from removing the batteries. A small Philips (+) screwdriver is needed to tighten and loosen the screw. This system operates at 4.5V and so requires three (3) AA batteries. The batteries are inserted by aligning the negative (-) terminal to the spring end side of the battery compartment following the embossed symbols.



Powering On/Off

To power-on your unit, first ensure that batteries are installed.

All units include power management capabilities to conserve battery life when forgetful users forget to switch the device off after usage.

The handheld coding block can be awoken from sleep by pressing and holding the Play start/stop button until the LED is glowing, and then releasing it. You may need to hold it for up to three (3) seconds before it responds by lighting the Status LED – now release the button. A start up tune is played as the device awakes.

Sleep mode is automatically activated after an extended period of inactivity (more than 5mins). If left idle, the code block will try to entice the user to interact by first blinking, and then playing a little ditty. After a number of rounds of this, it will then play a shutdown tune and suspend all activity. Every minute or so after this the unit will blink the LED to indicate it is in sleep mode.

Sleep mode can also be activated by the user when the code block is in idle mode by holding the start/stop button until the countdown finishes and the yellow LED goes out, and then releasing the button.

Note: When the batteries are weak, the power/idle indicator will glow yellow rather than red.

Batteries should last up to six months, however it is best to remove batteries if not using for a prolonged period.



Standard Coding KLIX

There are three basic type of coding KLIX, and a number of special function pieces to extend coding capabilities. Colours help to differentiate and group the pieces by function.

Output Actions



LED, or light emitting diodes, are KLIX that glow when your code runs. Each KLIX can be one of two colours depending upon how its connected. If its not the colour you expected when you run your code – simply flip it around! The LED can be made to flash using the modifiers.

The sound effects (SFX) KLIX will cause a noise to be sounded from your coding block's speaker.

There is a default sound when the KLIX is used by itself, however you can also select the desired sound from list by using a modifier before this KLIX in your code.

Note: whilst sound effects code is being played the code cannot be interrupted by pressing the stop button. Do not worry, SFX do not last very long...



	Selected Sound
-	Fire Engine Siren
1	Police Car Siren
2	Beeeeeep, Bop
3	Rocket / BlastOff
4	Beep, Beep, Beep, Beep
5	Long Buzz
6	Zap, zap, zap
7	Level Up SFX
8	Noise 1
9	Noise 2

Standard Coding KLIX

More Output Actions



	Selected Tune
-	Star wars
1	Indiana Jones
2	Inspector Gadget
3	George of the Jungle
4	Blue (da, ba, dee)
5	Jingle Bells
6	Barbie girl
7	Australian Anthem
8	Beverly Hills Cop
9	Frosty the snowman
10	Rudolf the Red Nose
11	Silent Night
12	James Bond Theme
13	Top Gun Theme
14	Gyruss Intro Theme
15	SMB goal

The tune (TUNE) KLIX will cause a short tune to be play from your coding block.

There is a default tune played when the KLIX is used by itself, else you can select the desired tune from the list by using a modifier before it in your code.

Tunes are played note by note. Some tunes can be quite long.

If you want to stop the playback, simply briefly press the start/stop button on your coding block.

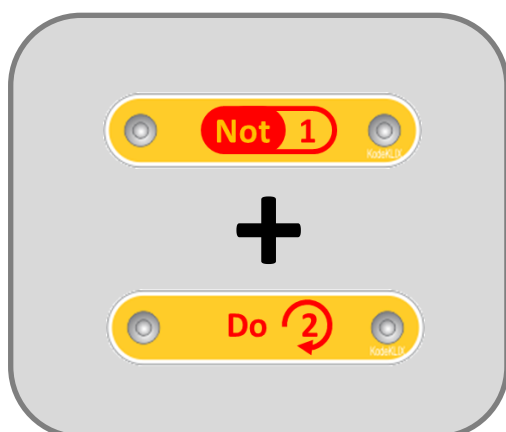
Note: the tune engine is monophonic, so do not expect a full symphony orchestra! Popular tunes released for early Nokia mobile have been translated for use with the KodeKLIX® coding block. The tunes are formatted as short ditties a few seconds long.

Some of these tunes are songs, some are movie themes, and some from retro games. Most young children will probably not recognise then – but it may bring teachers and parents a moment of childhood nostalgia.



Standard Coding KLIX

Modifiers



Numeric modifiers are just like numbers.

Used individually, they instruct the subsequent piece to behave in a particular way.

For example:

- If placed before an LED, they instruct the LED to blink the prescribed number of times; eg [2] = twice.
 - If placed before an SFX or TUNE KLIX the corresponding effect or tune will be requested;
- Numbers can also be sequenced to form sums. To play sound [3] you can also add KLIX [1] and [2]!

Note: do not include gaps between KLIX when performing sums as this gap resets the calculation sequence. Not all modifiers may be included in your kit.

The [1] KLIX also can act as a [Not] or opposite for some action KLIX. For example instead of light it can be used to check for dark (ie NOT light), or button “NOT pressed”.



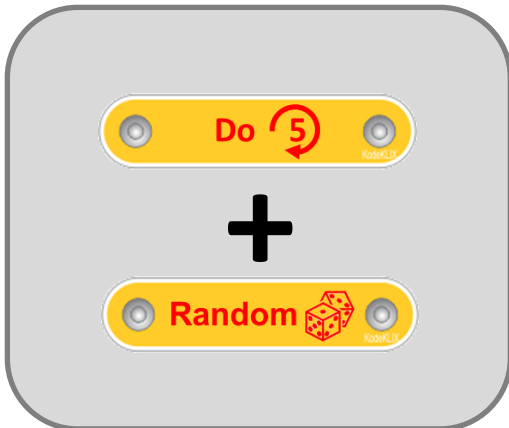
Standard Coding KLIX

Random Modifiers



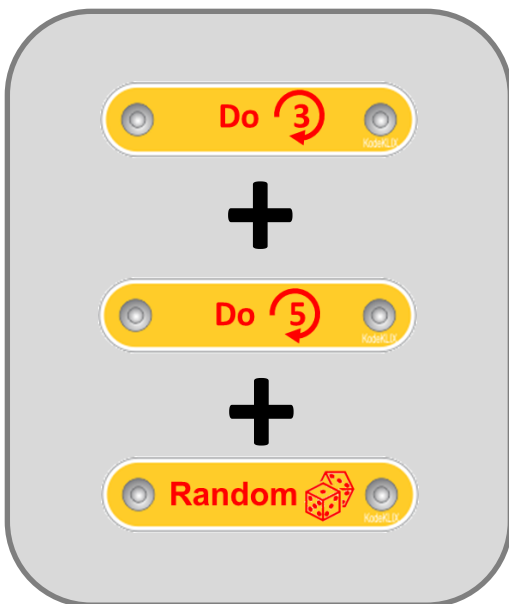
Random is a special number KLIX.

Random is like a dice; rolling values between 1 to 6. Like a real dice, sometimes numbers will be repeated.



When used in combination with the standard numeric modifiers it is possible to have random values that are in a range outside of 1 to 6.

For example, proceeding Random with a number like [5] then the corresponding result is [5]+[Random], resulting in values between 7 and 11 being generated.



Note: do not include gaps between KLIX when performing sums. Also, when using Random in summing operations, it is important to have this KLIX as the last in the sequence (as shown) in order for the arithmetic to be calculated correctly. This is because the dice roll is added to the preceding total.

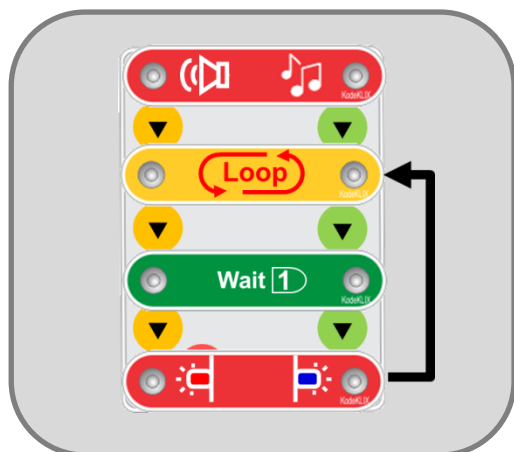


Standard Coding KLIX

Sequence versus Loops



Code is read and acted upon sequentially, unless a Loop command is included.



The Loop KLIX determines the return point for where to jump back to when there is no more sequential code to run. The Loop KLIX is placed at the beginning of the coding block that will be repeated; but that can be at any coding step.

Input Actions



Compared to output actions, inputs respond to action. This can be the passage of time or a user action such as pressing a button or tilting the device.



Since each input can respond decidedly different according to the how its used and the modifiers called up before it, the functionality of each input device will be specifically detailed on the next page.

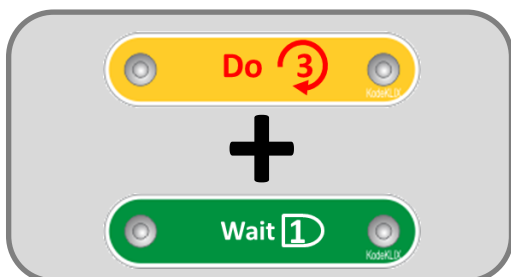


Standard Coding KLIX

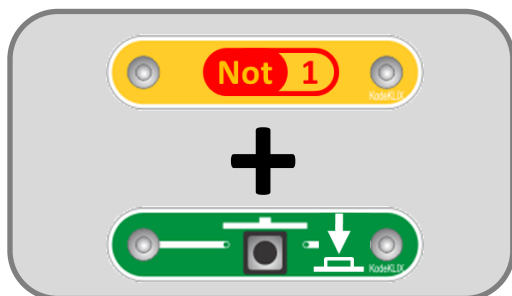
More details on Input KLIX



The Wait KLIX pauses the running of your code for the designated duration. The default is 1s (second). However the use of a modifier extends the pause for the desired duration. For example if a [3] modifier is used the delay is for 3s .



The button KLIX waits until a user interacts (or doesn't interact) with it. Without a modifier, code will wait until the button is pressed.



Using the [Not] modifier signals the code to do the opposite, ie not wait for the button to be pressed.



The tilt KLIX works in a very similar way to the button, except the "on" is when the code block is slanted in the direction of the arrow (eg left or right).



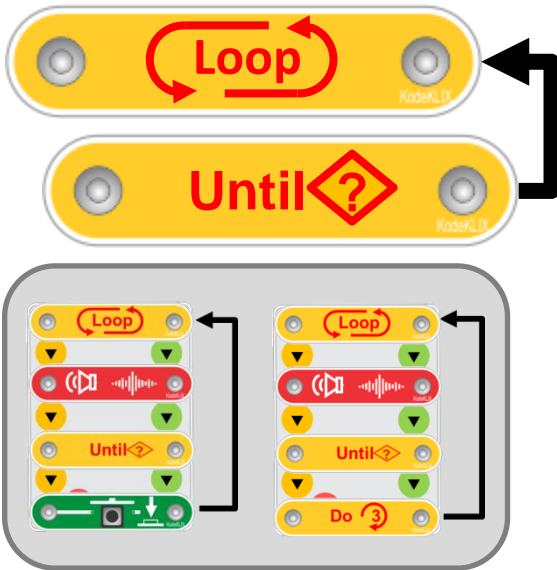
The light sensing KLIX will wait until sufficient light shines into the sensor. The level of light required can be adjusted using a modifier [2] to [5], or wait for darkness if inverted with [NOT1].

Note: the level of ambient lighting is set when the code is prepared whilst the status LED is yellow.



Advanced Coding KLIX

Conditional Loops



The Loop KLIX when used on its own defines the start of sequence which returns to this point when completed.

The Loop-Until construct provides a conditional end to the loop, based on either count (using a modifier) or input action (such as a button press, tilt or light sensing).

See the panel for specific coding examples of each Loop-Until condition.

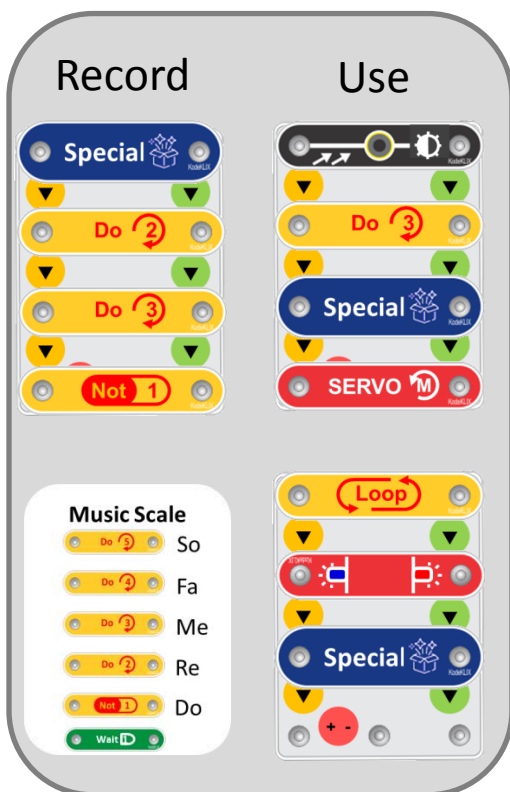
Special Functions



The Special KLIX is just that, Special, in that it can be used to record a sequence of modifiers for playback with just this one code piece being used. The Special sequence is directed to the subsequent action KLIX

The Special KLIX can be used multiple times and each time it will replay the recorded sequence. A modifier before the Special KLIX will repeat it.

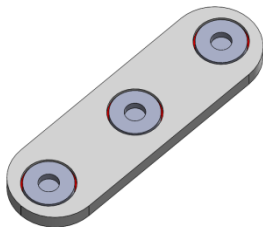
The Special sequence can be recorded by following the simple procedure of placing the Special KLIX in the first coding slot and then sequencing modifiers after it in the order of actions to be stored.



Advanced Coding KLIX

Most advanced action coding components need a third power snap. This snap is in the middle. Three snap parts have to be assembled in the correct orientation to work.

Smart Multicolour LED

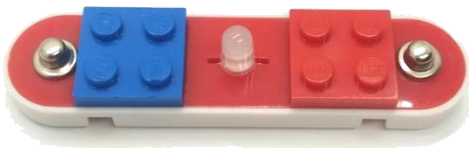


The RGB KLIX contains a special LED capable of displaying all three individual LED colours – Red, Green and Blue, and also select combinations.

The colour to display is selected by using the modifier code piece before it.

-	1	2	3	4	5	6	7	8	9	10
RGB flash	Red	Yellow	Green	Blue	Purple	White	R-B Flash	Y-Y Flash	Off	RGB Flash

Optional Brick Compatible Parts



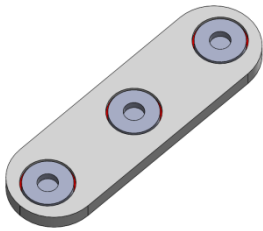
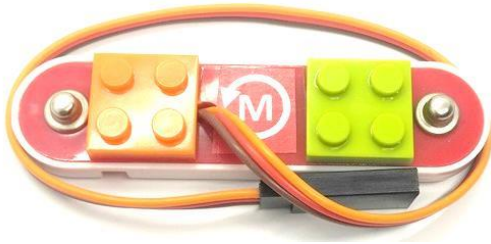
Coding KLIX with brick studs can be specially ordered or made at home. These components have brick plates bonded to the face plate allowing constructions to be assembled on the KLIX. Typically only the LED and Button components make sense to have in this brick compatible format.

See the construction examples for guidance on making your own parts.

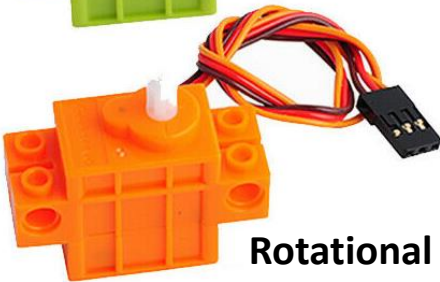


Advanced Coding KLIX

Smart Motor Servo



Positional



Rotational

Servo motors, including brick compatible versions, can be connected using our Smart Motor Adaptor. This special KLIX includes a female plug to connect with the motors.

One brick compatible brand available online is the GeekServo range.

To connect to the Smart Motor Servo KLIX, align the colours of the wired connector correctly and press together.

There are two types of servos:

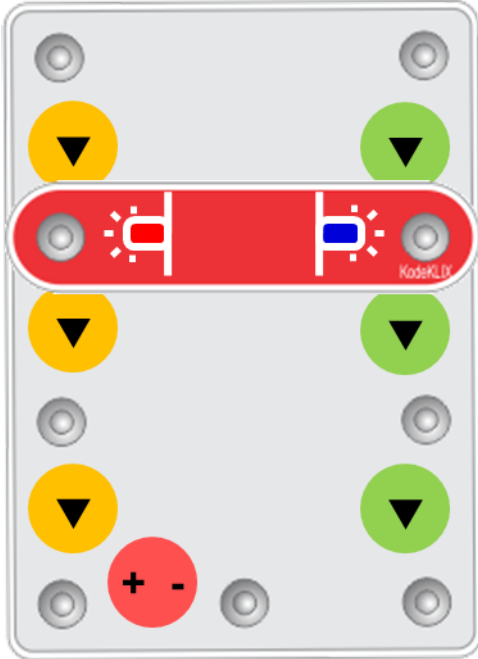
1. Positional (eg grey) which can move to an angular position
2. Rotational (eg green/orange) which rotate at speed

Modifier components can be used to adapt the function of the servo as shown in the table:

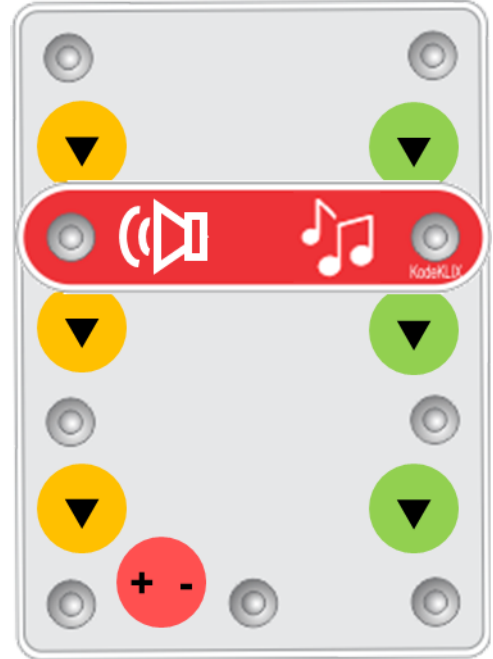
Modifier	Positional	Rotational
-	Full Forward	Fast Forward
1	Full Backward	Fast Backward
2	Backward	Slow Backward
3	Centre	Stop
4	Forward	Slow Forward
5	Full Forward	Fast Forward



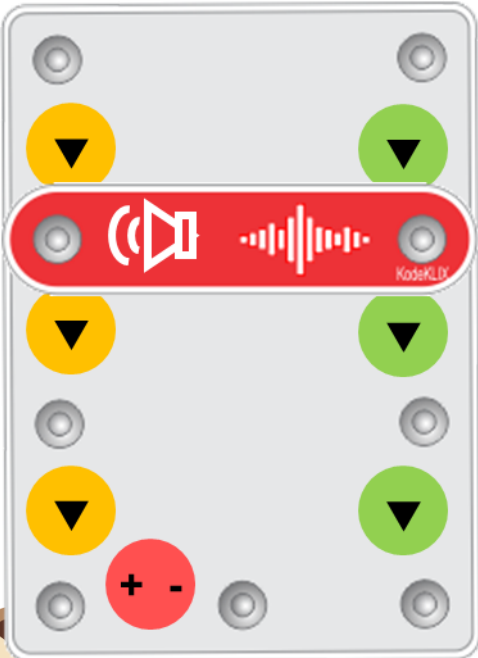
Coding Examples



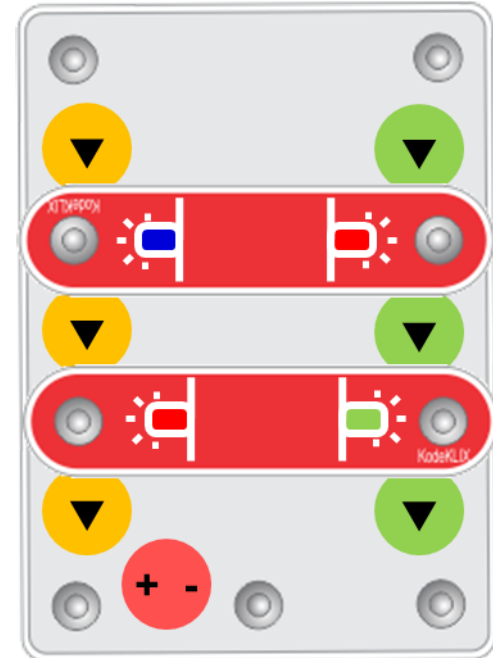
**Blink LED for 1s
and stop**



**Play default
Tune and stop**



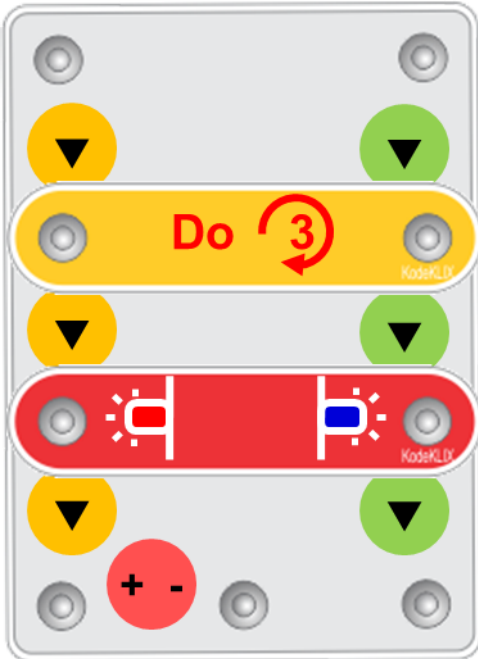
**Play default
Sound and
stop**



**Blink Red, Blink
Green and stop**



Coding Examples



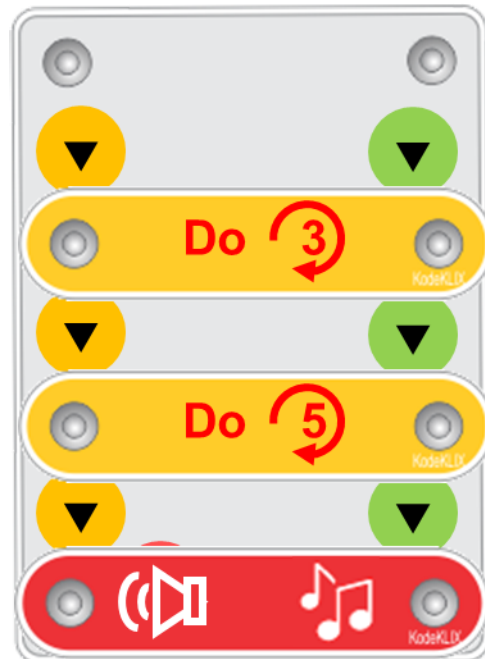
**Flash LED 3-times
and stop**



**Play Tune 3 and
stop**

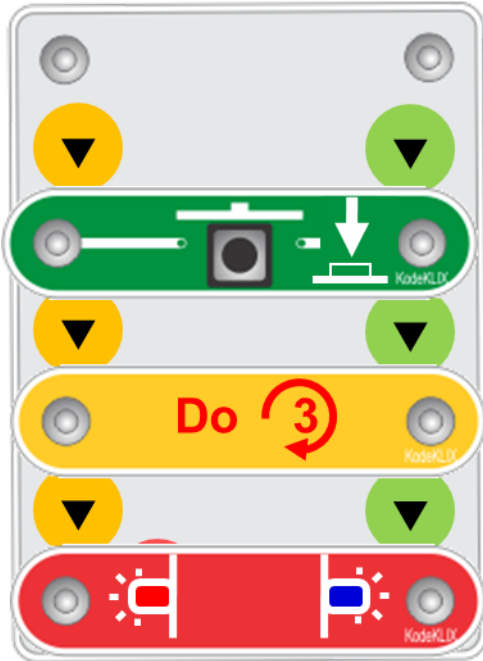


**Run Motor for 3s
and stop**

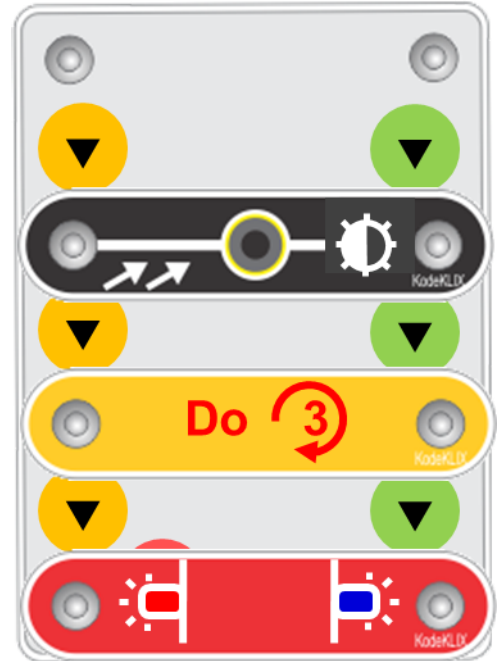


**Play Tune 8
(3+5) and stop**

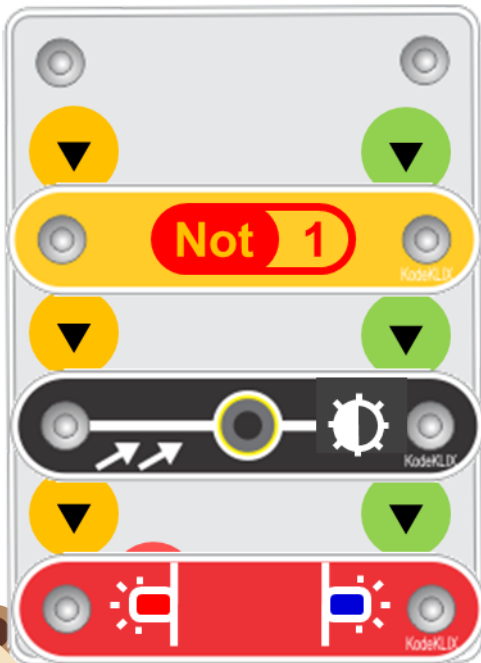
Coding Examples



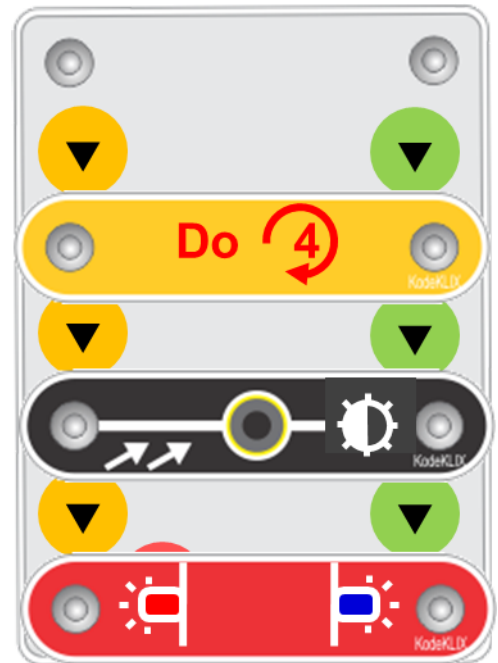
Wait for button, Blink LED 3-times and stop



Wait for Light*, Blink LED 3-times and stop



Wait for Dark*, Blink LED for 1s and stop



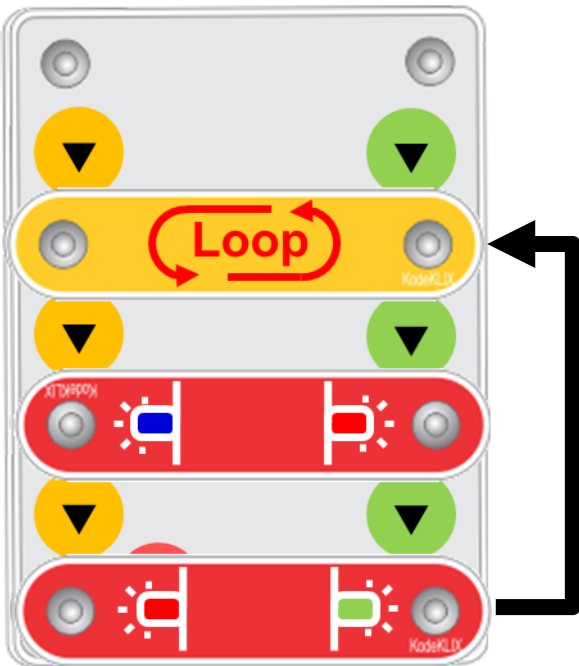
Wait for Brightness 4 Light*, then Blink LED for 1s and stop

** note: the light sensor adapts to the ambient levels each time the code is run*

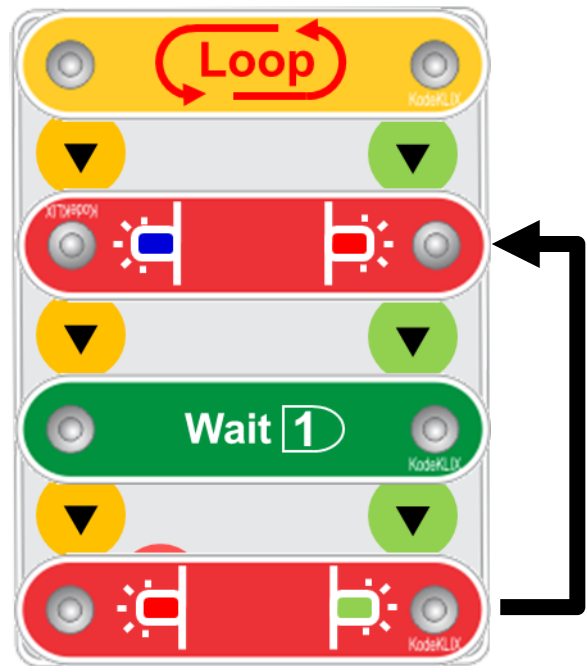


Coding Examples

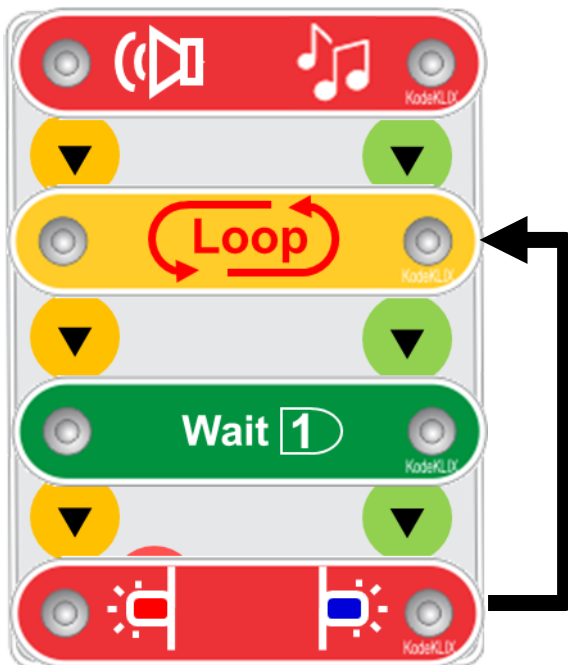
Simple Loops



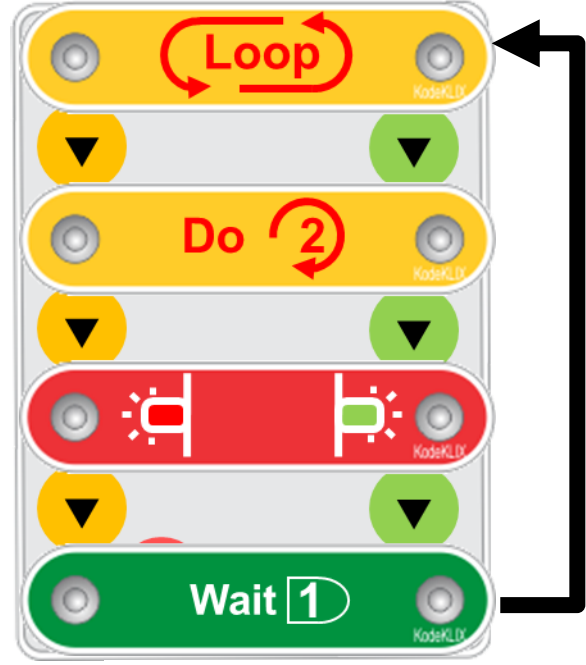
Alternating Red Green LEDs – 1 second each



Alternating Red 2s, Green 1s, repeat



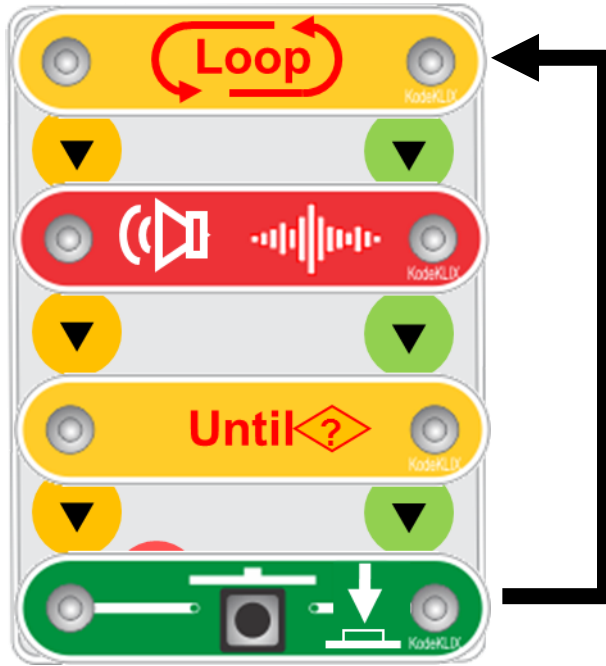
Default Tune, then enter loop; Blink LED at 1 second flash rate



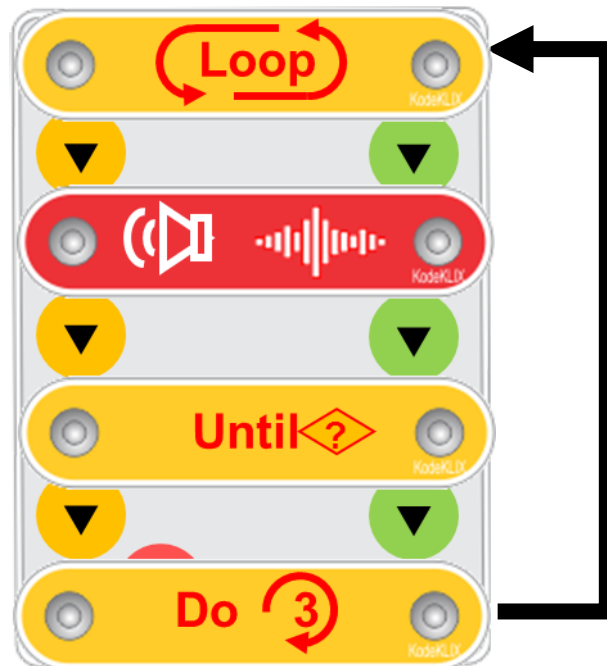
Blink LED twice, wait 1s, then repeat

Coding Examples (advanced)

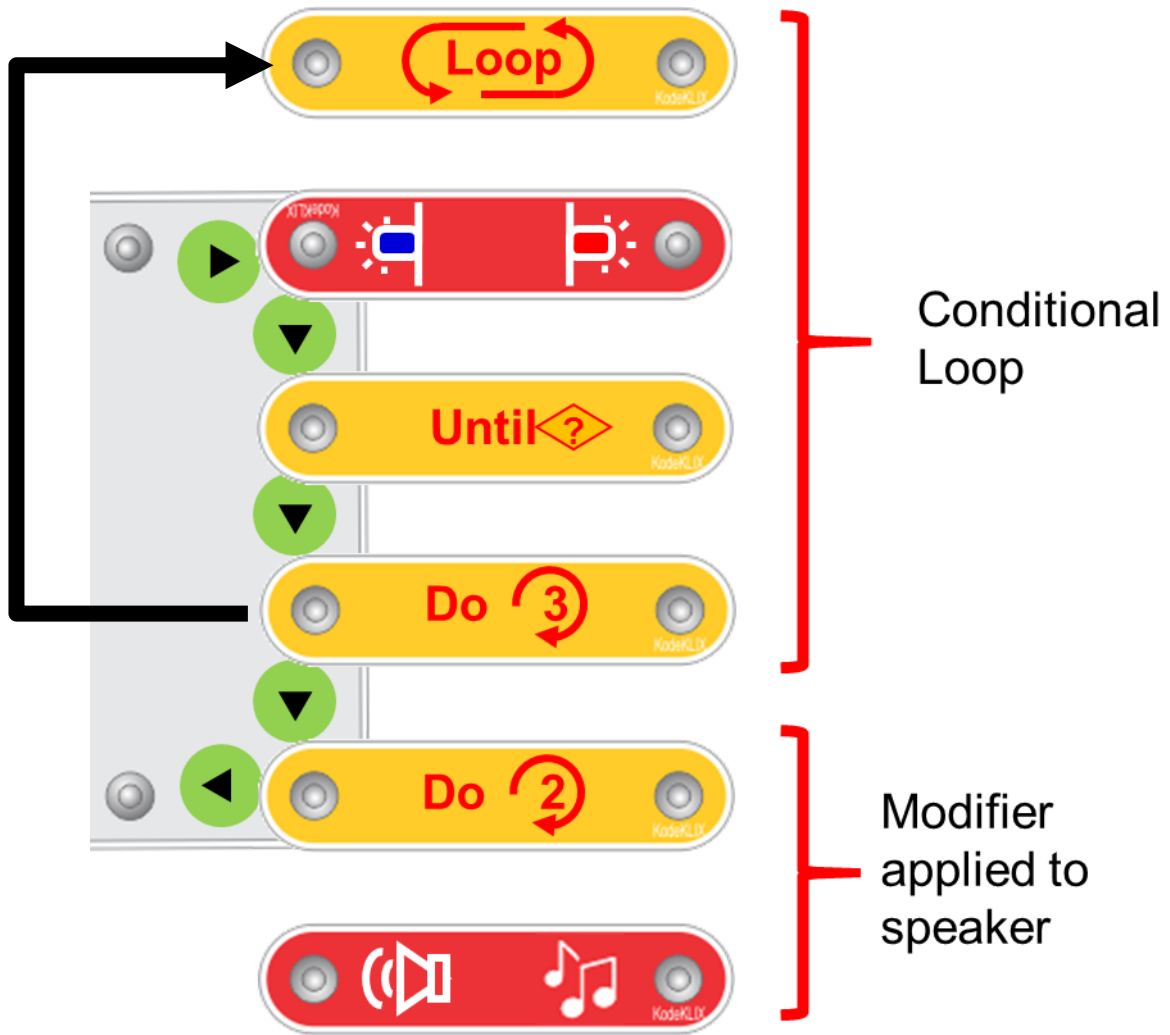
Play Sound, until button pressed, then end



Play Sound, 3 times, then end



Coding Examples (advanced)



**Blink LED on/off,
UNTIL 3x LOOPS completed,
play Tune 2,
then end**

Conditional Loop-Until



Coding Examples

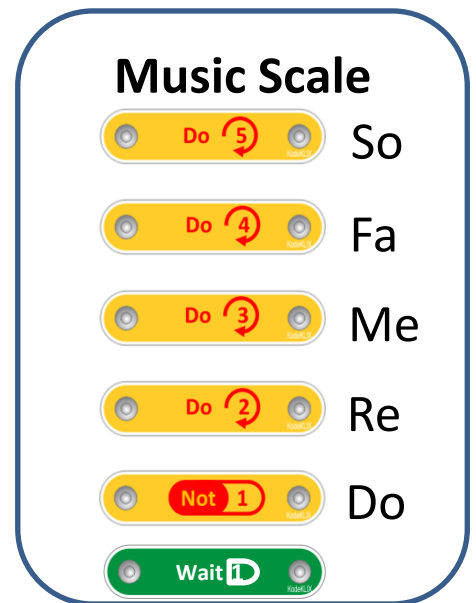
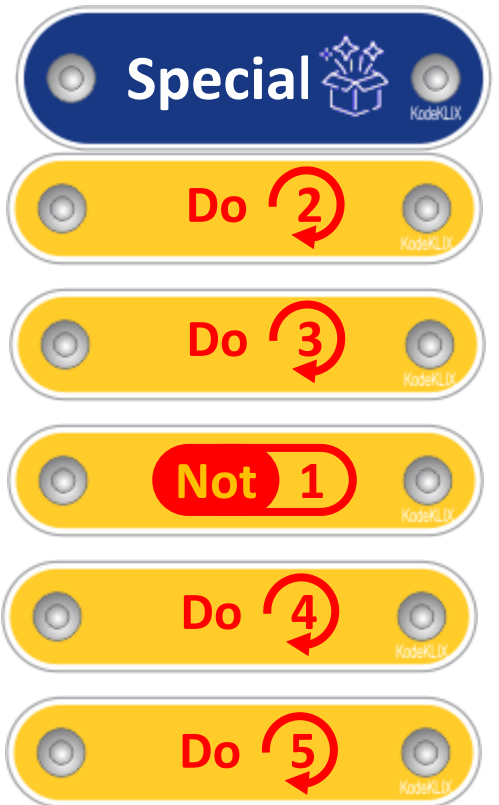
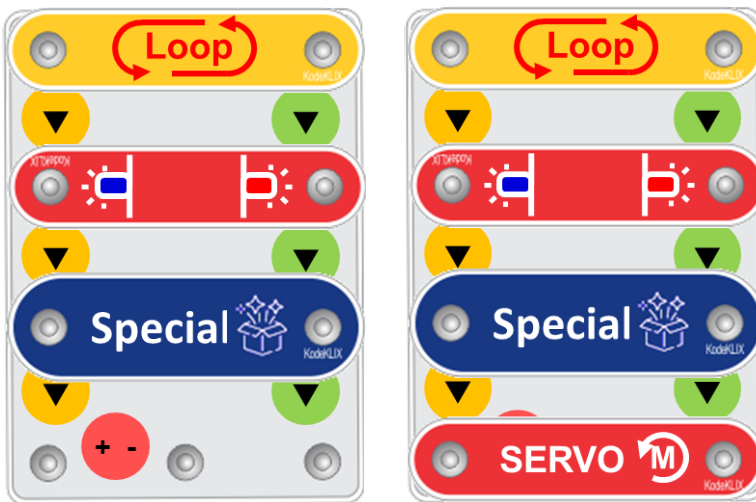
Two step process

1) Defining the subroutine

By making the first coding element the Special KLIX*, the modifiers listed are used to define the user sequence. The modifiers represent actions or musical notes (see scale).

2) Using the subroutine

When the Special KLIX is used in any other location in the sequence, it is utilised like any other code.



If the Special KLIX is followed by an output action, the modifier sequence is applied to that piece, eg to operate the servo. If no action is specified, it proceeds to play notes through the built-in speaker. Special KLIX have no effect on input actions, and default to playing notes.

* your kit may have more than one Special KLIX, but they all share the same sequence definition.

Coding Examples

Subroutine Actions



SPECIAL is blue in colour

- Without action, defaults to Sound



Sound Subroutine

- Modifiers used to select 1-of-5 notes based on sequence; Wait is pause, NOT is 1



Servo Subroutine

- Modifiers used to select 1-of-5 speed/dir or angle settings based on servo type attached



LED Subroutine

- Modifiers used to select 1-of-5 blink or flash rates



Smart Multicolour LED Subroutine

- Modifiers used to select 1-of-5 to select colour



The use of a modifier before the Special KLIX acts as a multiplier, making the Special sequence repeat the indicated number of times (see example).



Runs this Special sequence 4 times





28

Coding Do's and Don'ts

To make your code all you need to do is snap the KLIX in the order you want things to happen. There really is no right or wrong way, you just list them out in sequence, from the top to the bottom of your coding system.

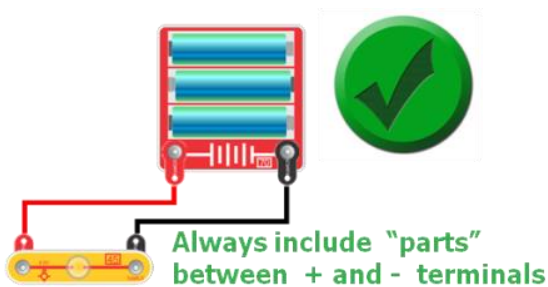
The internal logic engine in your coding block will endeavour to interpret your code, and execute it. However, like all logic systems – artificial or natural – there are some small ground rules to ensure what you get is what you expected.

It does not matter whether:

- KLIX pieces are placed upside down or right side up
- you start code at the first snap or at later snap (see Special*)

It **does** matter:

- if KLIX pieces are stacked (the code may not work as intended)
- mathematically, that you place Random after the other modifier numbers; if not the Random KLIX will be ignored;
- that three-snap advanced KLIX (like the servo adaptor) are aligned “green-to-green” in order to work as intended;
- that fresh batteries are used for best servo motor performance. These motors need at least a 3x AA battery pack connected (the handheld unit has this as standard). Replace your batteries if the power / idle indicator LED glows yellow rather than red.
- if the Special* KLIX is at the first snap location since this will redefine a stored subroutine sequence



Troubleshooting

Coding and logic issues

- Waiting for light input – when the code is checked, it uses the ambient light level as the current state. If you move to a different room, or go outside the ambient level will change. Restart your code when in a new environment.
- Special/Subroutine KLIX not working:
 1. If installed as the first KLIX, then its in define mode
 2. Possibly its been defined with no values; redefine
- At the code checking step I hear three error beeps; you are possibly using a KLIX which is not compatible with your model or version of the KodeKLIX® coding block.
- I cannot stop my code! Press the play button again to stop.

Component issues

- Wake-up from auto-off; hold the play button longer – it can take up to 3 seconds check the button if in deep sleep
- Servo KLIX not working:
 1. Three snap KLIX must be installed correctly orientated; green snap to green on the coding block.
 2. The connector wires to the motor must be aligned colour for colour; brown to brown, etc.
 - The RGB KLIX beeps when installed and code wont run; being a three snap KLIX it is probably installed back-to-front.
 - Other regular snap electronics parts **do not** work with the code block; not detectable!



Construction Options

Decorating and customising your Code Block

The KodeKLIX® included in your *Coding with Oscar* kits are made from durable ABS plastic. Your coding block can be decorated with stickers to give it that personal touch. The hand-held module in Kit 1 features the elements of a critter – with LED for eyes, a speaker as a nose, and exposed circuit as the tummy. You may use markers to draw other features such as ears, whiskers or a tail to further customise its appearance if that is what you desire and permit.

Arts and Craft approach

The robust construction of the coding block means you can also built into models made from craft materials such as cardboard, wood, felt or other craft materials. Its best to use tape to attach any items as some glues are permanent. Note also that any stickers on the coding KLIX may be damaged with the use of tape, so its best to limit the use of tape to just the coding block itself.

Brick Constructions controlled by Code

Construction using bricks such as Lego® and Duplo® branded products are ideal because of their re-usability and endless creative freedom. The next few pages show techniques for incorporating the code brick into such constructions. Your code can then interact or control your brick model.

Electronic Circuits

Coding with Oscar is a stepping stone to more detailed coding and electronics experiences. Kit 2 takes the first step in this journey by introducing the simple construction of *Oscar's first Coding Computer* and then exploring some of the fundamental electronic principles of circuits.

© CoplinCorp 2023. KodeKLIX® is a registered trademark.

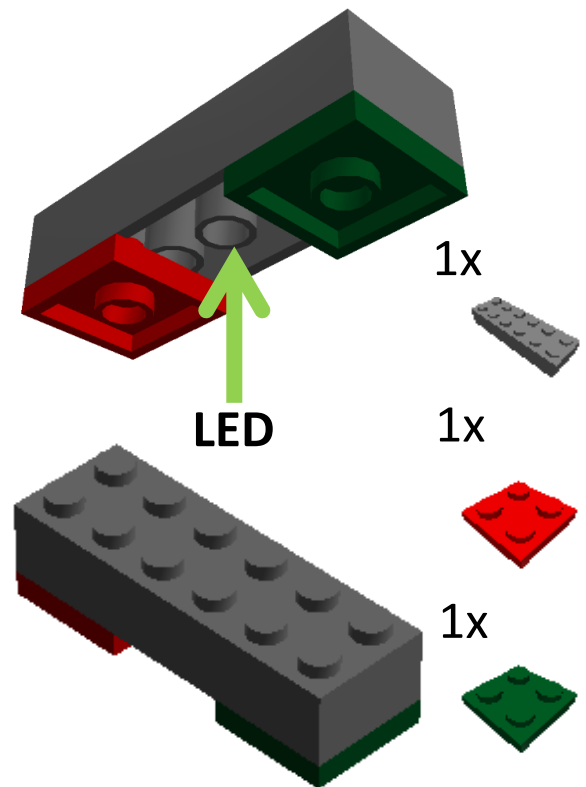


Brick Construction Techniques

Standardised brick spacing

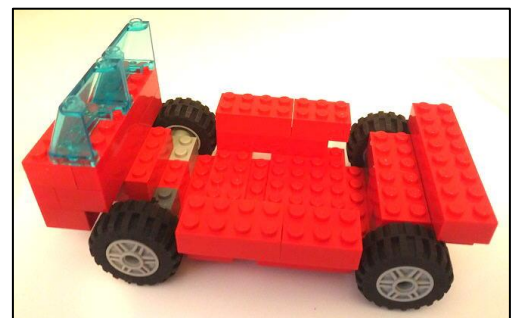
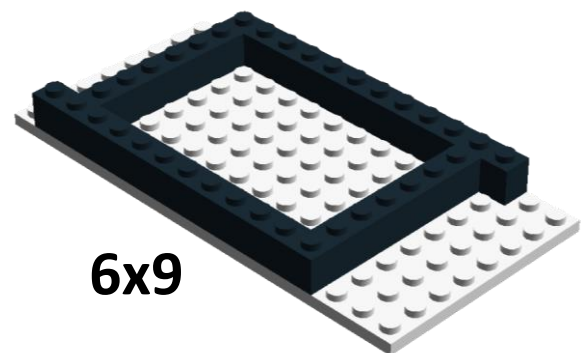
The spacing between brick is set to be centred on the LED. The LED fits inside the brick peg hole, and directly fit to the 2x6 form.

In the DIY example shown, the red and green plates can be bonded to a snap component using superglue and a simple jig made with the indicated Lego bricks.



Battery Box Size

The battery box on the rear of the Kit 1 coding block is conveniently sized to align with a brick frame 6 studs wide and 9 studs long, with a minimum wall 1 row tall. This frame will support your coding block and prevent it from sliding off your models. The base can also be integrated into your models as shown.



Brick Construction Techniques

Build on top of your code

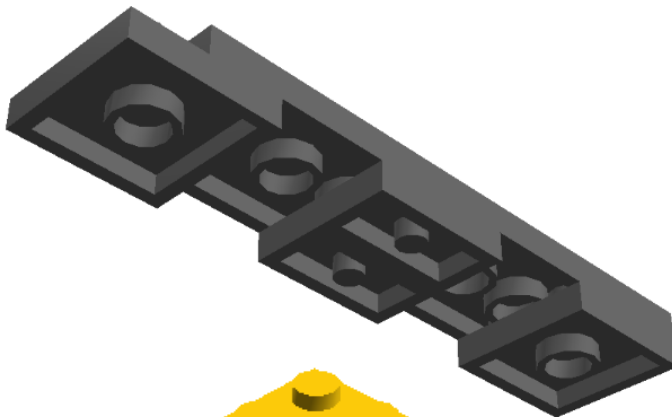
Bricks can also be connected across snap components, but you need to remember the stud spacing aligns only every second KLIX.

In between you will need to use the construction shown below to provide the half stud offset. The 1x2 knob used is quite common.

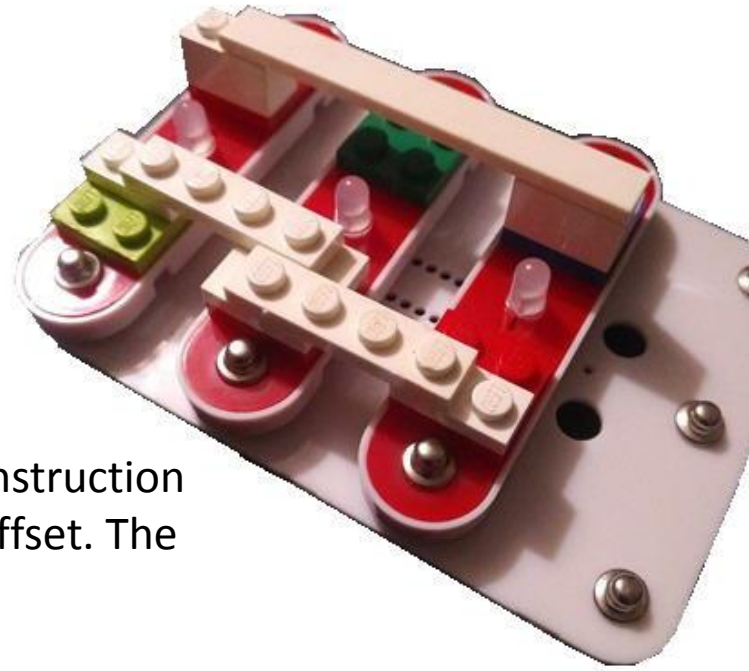
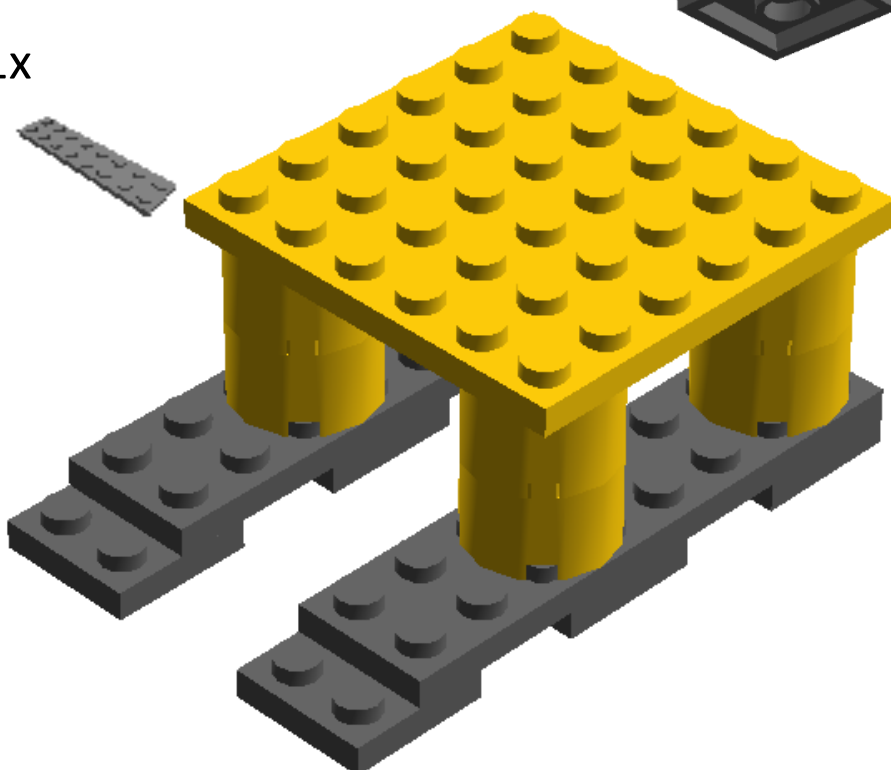
2x



1x



1x



The plate structure shown can be built in any colour brick and extended to suit the number of snap elements that you need to span.

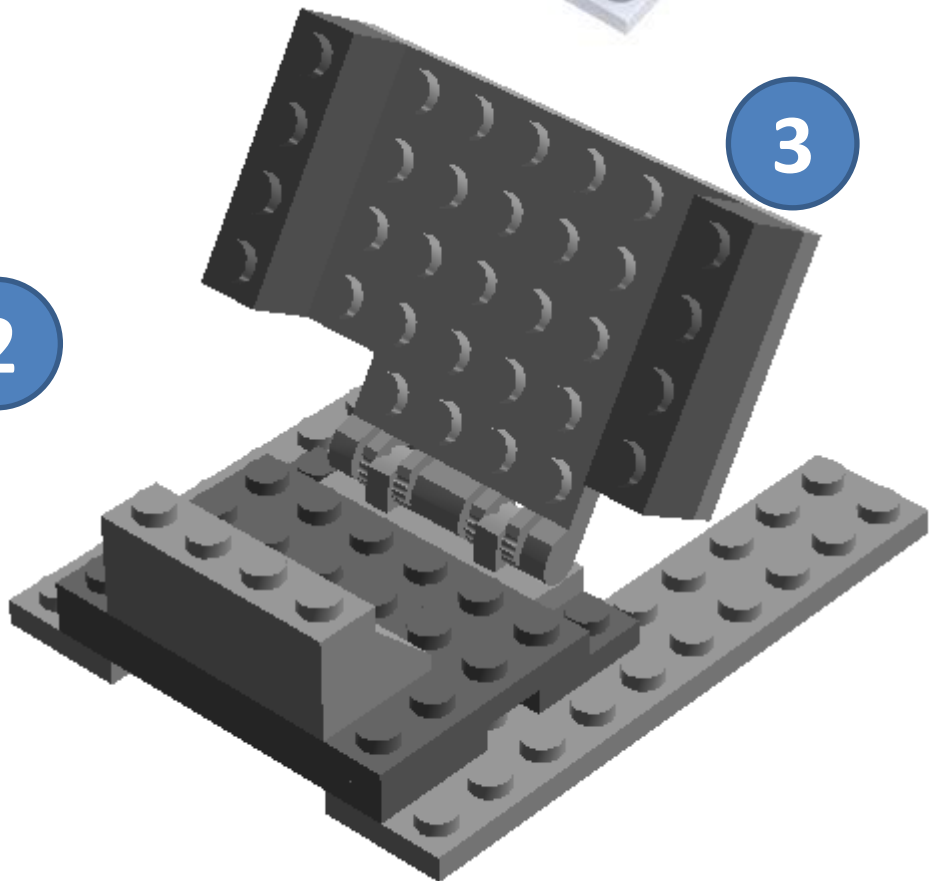
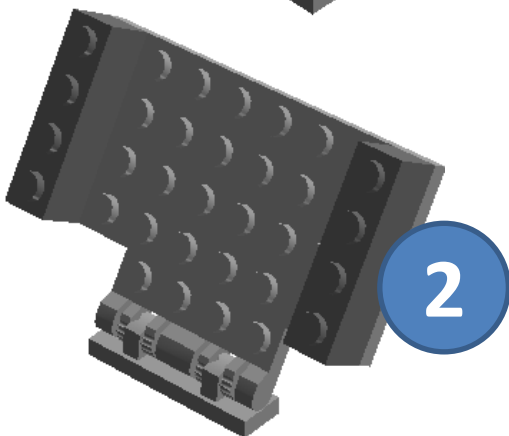
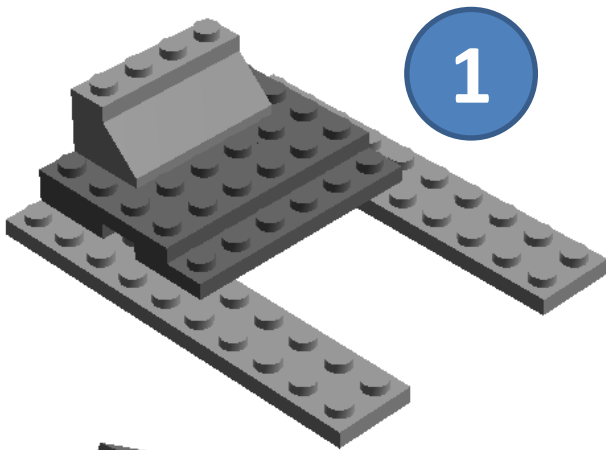
On top of this, your regular models can be assembled – including features for lighting elements to glow through.



Brick Construction Techniques

Adjustable brick stand

An adjustable stand for your coding block can be made using construction bricks. Just follow these three steps. Any colour bricks you have can be used.



Brick Construction Techniques

Model Gallery for Inspiration

Below are a range of brick model ideas to inspire the creativity of your children. You can use both small and large form bricks.



Electronic Circuit Projects

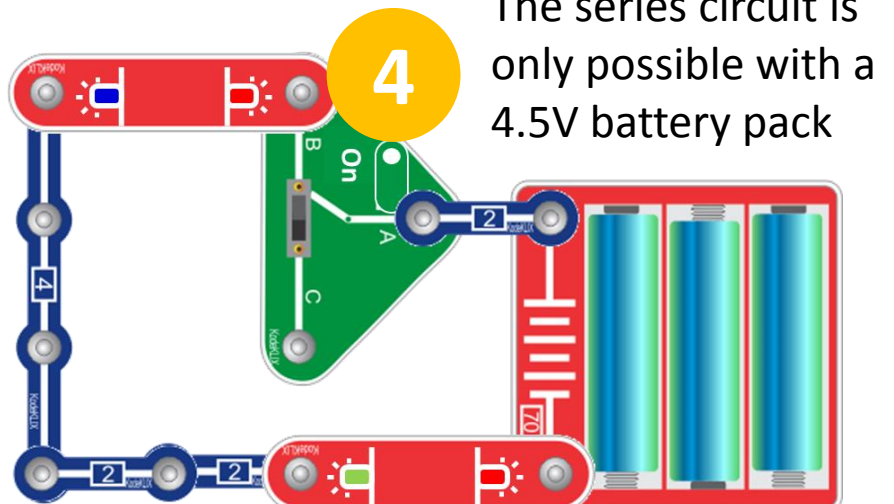
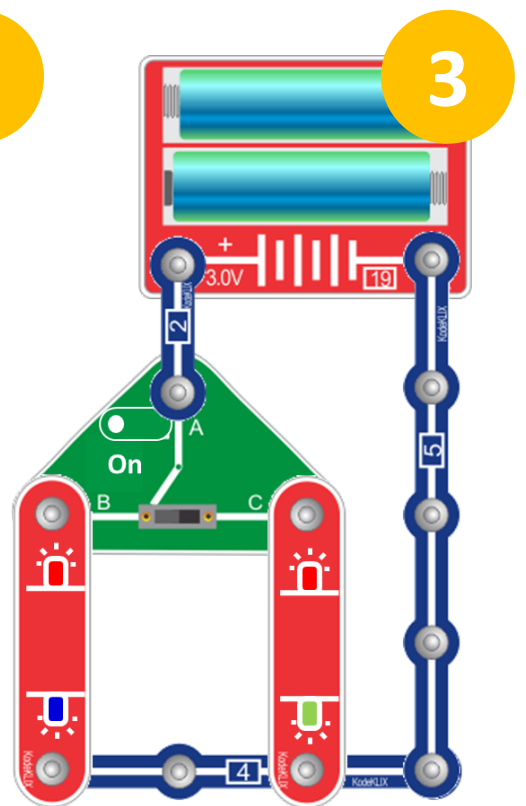
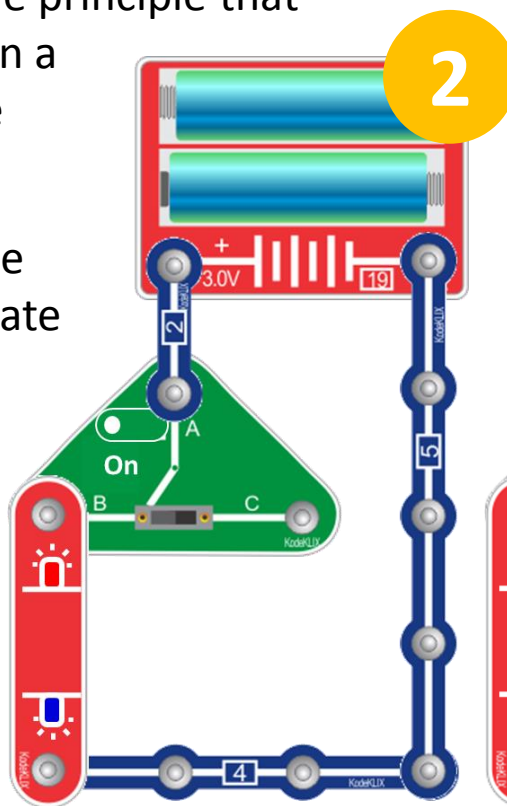
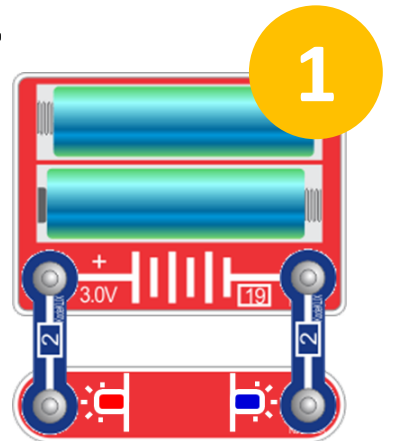
Kit 2 – Oscar’s First Coding Computer

Not all coding KLIX in the kit can work as electronic components. This is because many do not contain an active part. The circuits show here are examples of the individual electronic circuits that can be assembled.

Circuit (1) shows the principle that current must flow in a loop. Current is the flow of electricity.

In circuit (2) the role of a switch to regulate the flow of current is introduced.

Circuit (3) adds a parallel branch to the circuit in (2).



The series circuit is only possible with a 4.5V battery pack



Playing Games - Overview

Built into each *Coding with Oscar* coding block is a logic engine which not only executes the code that is created with the coding KLIX, but also scores it.

The coding engine uses some simple Artificial Intelligence, or AI, to work out which features demonstrate more advanced code construction, greater user interactivity or better engagement. These factors are combined to produce a code score of between 0 and 5. The code score is relayed to the user through either the multi-coloured LED eyes or by a more advanced display.

This code score can then be used in a number of interactive board games developed for *Coding with Oscar*.

Code scoring mode is only activated when you use the system in game mode. Game mode is invoked by pressing the play button with no pieces on the coding lattice. After the countdown sequence is initiated, one of two tunes is played for approximately 20 seconds.

This is the time you have to create some code. At the end of the timer, the code is scored and the result presented. The two tunes allow for two players, 1 and 2.

Be warned though... if you think you can use the same code each turn then you are wrong!. The AI engine will detect this and penalise your score!!!

JUMPS 0 1 2 3 4 5

Neural Network Zone

START **Finish**

Rules

- Use the code timer whilst you create your code
- When the time runs out a score colour is provided
- The colour determines the number of jumps
- Take the short-cuts

One Player scoring

- Count the number of moves you take to finish
- Table turns your moves into a score

Two Player winner

- First to finish is the better coder!

Coding Score Card

Moves	1	2	3	4	5	6	7	8	9	10	11	12	+		
Score			100			90		80		70		60	40	20	10

Coding with Oscar © Coplin Corp 2022. KodeKLIX® is a registered trademark. Kode KLIX Coding, Electronics and Fun! Ladder Maze Game 1.0

Beat the Bugs and don't Crash the System! **Score**

																10
																20
																30
																40
																60
																80
																1000
																900
																800
																700
																600
																500
																400
																400
																200
																200
																100

Rules

- Use the code timer whilst you create your code
- When the time runs out a score colour is provided
- Use the colour to work out which direction to move

Player scoring

- Move one step each time; if you Crash miss a turn
- Take the Bonuses, watch for Bugs
- The faster you finish the higher the score and the better your code

Coding Score Card

Moves	1	2	3	4	5	6	7	8	9	10	11	12	+		
Score			100			90		80		70		60	40	20	10

Coding with Oscar © Coplin Corp 2022. KodeKLIX® is a registered trademark. Kode KLIX Coding, Electronics and Fun! Maze Map Game 1.0



Playing Games - Scoring

So what code scores better than other code?

Here's a few insights into how the AI engine evaluates your code:

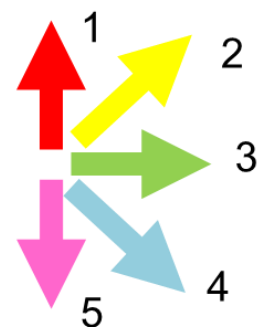
- Code length – whilst long code is not a great thing on its own, it shows that the user is capable of longer sequences.
- Code loops – code which has repetition demonstrates understanding of a fundamental computing format.
- Code interactivity – code requiring a user to participate is rewarded. In a world where automation is the norm, this element rewards user engagement.
- Code extensions – functions, use of maths and conditional logic are all rewarded as demonstrating developing skills.

Remember though, that top scores are limited to 5 and so you may not see the subtle nature in which the AI does its job. So what is the point of a high score when it comes to game play?

The interactive games are base on two play principles:

- Linear mazes, think Snakes and Ladders. The higher the score, the faster the your play and likelihood to win.
- Map based mazes, where your score is a directional vector. The higher your score the more direct your path to the finish.

Both games could be played with a dice, but scoring user code gives them control over their progress and destiny.



38

Playing Games - Making

Pre-made games and maps are available from the Coding with Oscar website for download and printing.

Any token piece can be used to track progress during the game play. For example children can use a coin, a Lego mini-figurine or something that they may have made from modelling clay or 3D printed in class.

Templates for both game formats have been included in subsequent pages of this guide. These can be printed and modified by children, parents and teachers to create your own themed gaming experience.

- The linear maze “Snakes and Ladders” themed game can be as long or as short, as easy or as complicated as you want. It may only contain “ladders” or only contain “snakes”. Remember, these can also be time portals and trap doors; do not let imagination and creativity be stifled!
- The map based maze game can equally be customised. To assist with this, there is a label sheet which can be directly applied to the map template to quickly create a new game field. Remember to be fair. A game needs to be fun, and include a possible path to take to the finish.

Templates and games you create are free for you to share!

JUMPS	0	1	2	3	4	5
-------	---	---	---	---	---	---

Linear Maze

Rules


- Use the code timer whilst you create your code
- When the time runs out a score colour is provided
- The colour determines the number of jumps
- Take the short cuts
- One Player scoring**
- Count the number of moves you take to finish
- Table turns your moves into a score
- Two Player winner**
- First to finish is the better coder!

Coding Score Card													
Moves	1	2	3	4	5	6	7	8	9	10	11	12	+
Score	100			90			80	70	60	40	20	10	

Beat the Bugs and don't Crash the System!										Score
Map Maze										10
										20
										30
										40
										60
										80
1000	900	800	700	600	500	400	400	200	200	100

Rules

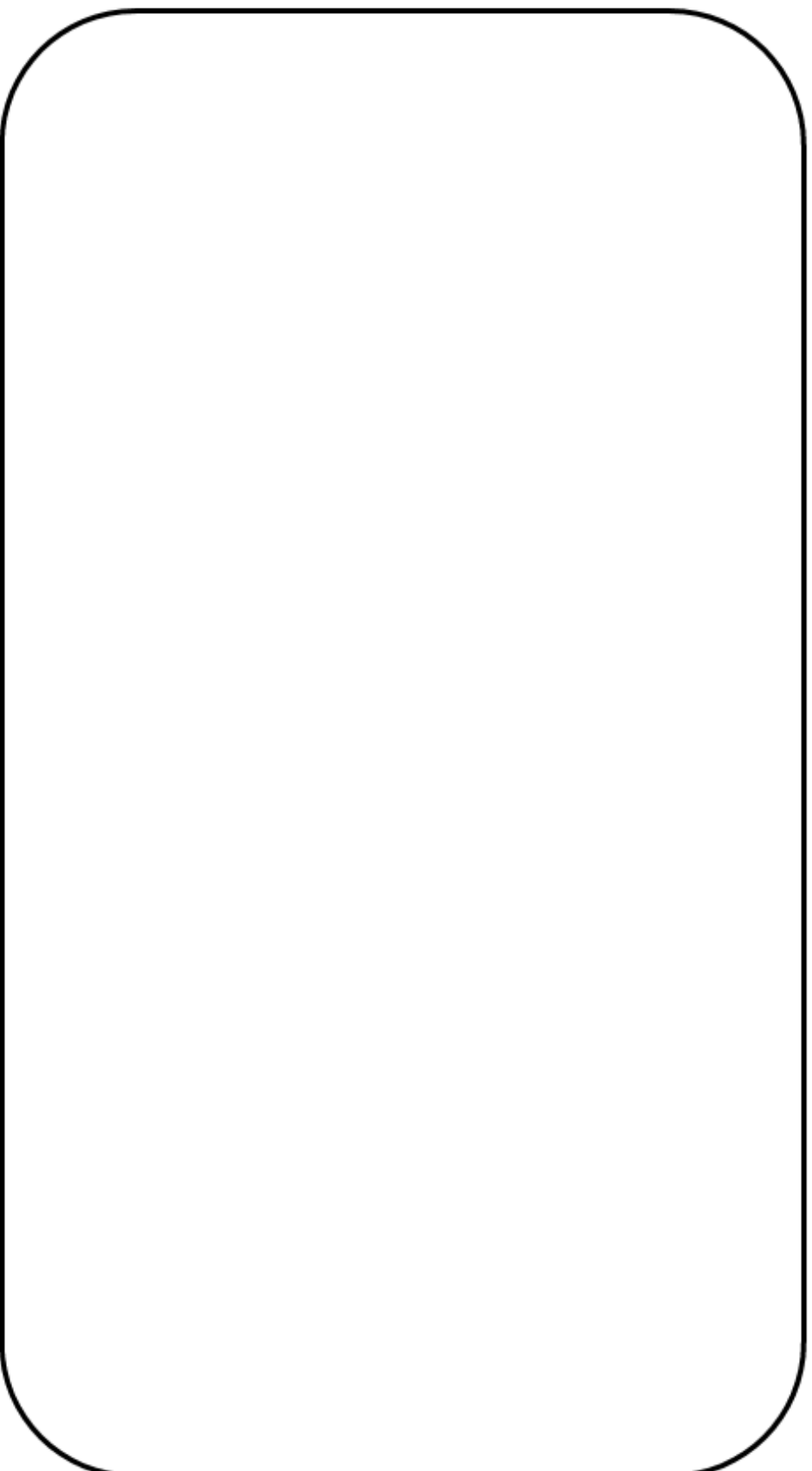
- Use the code timer whilst you create your code
- When the time runs out a score colour is provided
- Use the colour to work out which direction to move



- Move one step each time, if you Crash miss a turn
- Take the bonuses, watch for Bugs
- Player scoring**
- The faster you finish the higher the score and the better your code



JUMPS	0	1	2	3	4	5
-------	---	---	---	---	---	---



- Rules**
- Use the code timer whilst you create your code
 - When the time runs out a score colour is provided
 - The colour determines the number of jumps
 - Take the short cuts
- One Player scoring**
- Count the number of moves you take to finish
 - Table turns your moves into a score
- Two Player winner**
- First to finish is the better coder!

Coding Score Card																		
Moves	1	2	3	4	5	6	7	8	9	10	11	12	+					
Score	100					90							80	70	60	40	20	10

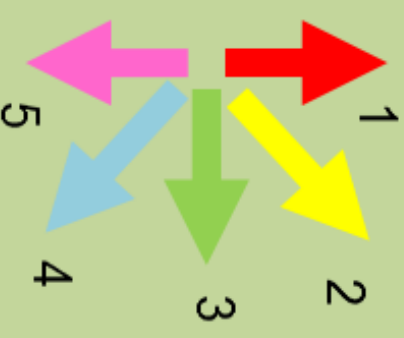


Beat the Bugs and don't Crash the System!

Beat the Bugs and don't Crash the System!											Score
											10
											20
											30
											40
											60
											80
1000	900	800	700	600	500	400	400	200	200		100

Rules

- Use the code timer whilst you create your code
- When the time runs out a score colour is provided
- Use the colour to work out which direction to move



- Move one step each time; if you *Crash* miss a turn
- Take the *Bonuses*, watch for *Bugs*

Player scoring

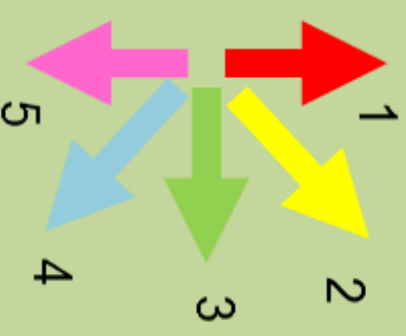
- The faster you finish the higher the score and the better you code

Beat the Bugs and don't Crash the System!

START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash
START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash
START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash
START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash
START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash
START	Bonus ▼▼	Bonus ▼▼	Bonus ▼	Bonus ▶	Bug ▶	Bug ▶	Bug ▶	Bug ▶	System Crash	System Crash

Make your own

- Cut out and stick the different field elements to your own game map.
- The more bonus squares you have the easier to score high; the more bugs the harder the game is.
- Remember, you can use more elements than each row has, but try to keep your game fun as well as challenging.
- Token moves as per the arrows:



Curriculum Linkages

Coding with Oscar introduces concepts which are fundamental foundations for logic and computer coding in later years. These include the following alignments:

KLIX	Code and educational outcomes
Actions	Inputs and Outputs
Loops	Sequential and repetitive code
Wait	Delays and timing basics
Modifiers	Numbers and simple sums
Modifiers	Function parameters; indexing from tables
Random	Chance and probability
Until	Conditional logic; waiting for decisions/actions
Special	Subroutines and repetitive code

Because of the fundamental difference to robot based learning systems, by using coding pieces that resemble block based coding representations used later in school (eg Blockly, Scratch), the cross-over to these screen-based systems is expected to be easier.

In addition to direct logic based learning, there are intentional alignments to other early learning aspects. Simple maths was noted in the table above. Literacy is touched upon not only by the interactive stories but also through the use of word based names on the coding KLIX – many KLIX have a word name with a unique first letter in its most common phonic usage, eg Loop, Do, Until, Random. Whilst not all children will remember the spelling of the word, the first letter will be readily recognised.

There is also a strong play and discovery based approach to the lesson plans; relying on hands-on interaction to cement the actions described in the stories.



Curriculum Linkages

Reviews with professional educators* has allowed us to delineate the typical age appropriateness of the *Coding with Oscar* kit levels.

Whilst every child will be different in their development status, and various factors can affect this from home to school, the table below may serve as a guide to teachers and parents as to the average age of the cohort's ability to work with coding systems and computers in general. It is for these reasons that screen-free based coding methods are more effective at this early age group.



Early Age Learning/Doing Capability

Feature	5-6yr Old	7-8yr Old	All age	Comment	Can	Some	Can't
					✓	•	✗
Snaps	✓	✓	✓	Hands on construction			
Story telling	✓	✓	✓	Learning by through listening			
Free play building	✓	✓	✓	Free form creativity			
Pictorial copy builds	•	✓	•	Follow "Lego" style builds			
Actions	✓	✓	✓	Understand consequences			
Sequencing (linear)	✓	✓	✓	Follow steps in a straight line			
Sequencing (circular)	•	✓	•	Follow steps in a circle			
Loops	✓	✓	✓	Repetition of steps			
Decisions	•	✓	•	Conditional moves/changes			
Indexing Information	✗	•	✗	Using one piece of info to find another			
Board games / rules	•	✓	•	Quite dependent on rules/genre			
Computer Usage	•	•	•	Ability to use a PC based app readily			

5-6years is guidance only; beginner or starting level.

7-8years is guidance only; experienced or advanced level.

8years plus should be considered for tablet or app based coding systems; but depends on prior experience.

** Thanks to key professionals from the Association of Independent Schools – WA, and other schools, parents and professionals who reviewed beta releases of this kit.*



Curriculum Linkages

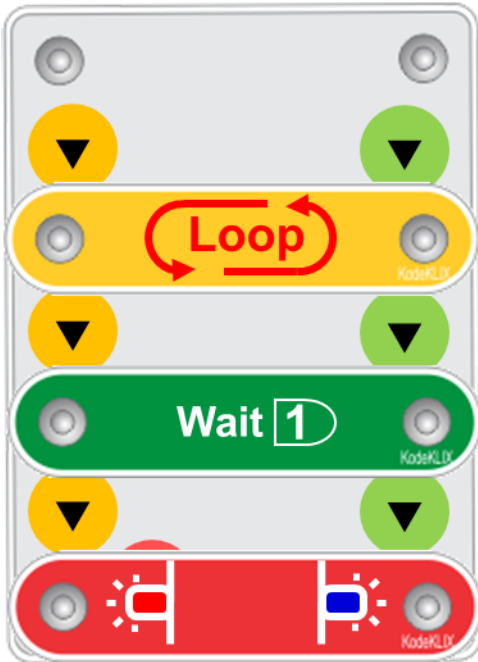
Science	Technology	Engineering	Maths
<ul style="list-style-type: none">• Colours• Light / dark• Sound	<ul style="list-style-type: none">• Sequential and repetitive coding• Logic• Function and parameters	<ul style="list-style-type: none">• Inputs and outputs• Actions	<ul style="list-style-type: none">• Counting• Numbers and Sums• Chance and Probability
<ul style="list-style-type: none">• LED KLIX• Light sensor KLIX• Sounds KLIX• Tune KLIX	<ul style="list-style-type: none">• Loop and until conditions• Indexed lookups• Delays & Timing	<ul style="list-style-type: none">• Construction• Design• Automation and control	<ul style="list-style-type: none">• Number KLIXs• Counting blinks• $[1]+[2] = [3]$• Random KLIX





46

Coding Comparisons



Different coding languages shown below are compared to the approach used with Coding With Oscar coding KLIX.

Script based languages need very specific syntax and structure.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize
  digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  delay(1000);           // wait for a second
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on by making HIGH
  delay(1000);           // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making LOW
}
```



```
## ---- Imports ---- ##
import time
import board
from piper_blockly import *

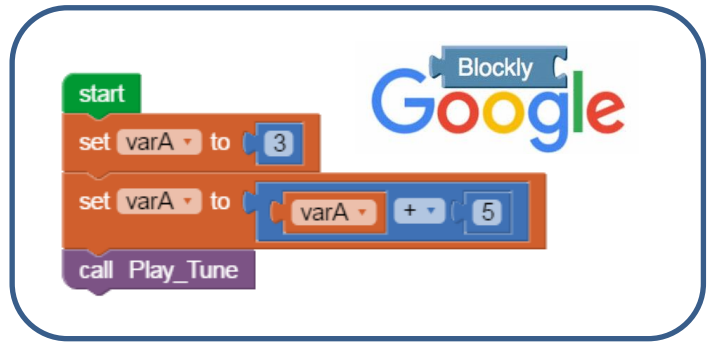
## ---- Definitions ---- ##

GP0 = piperPin(board.GP0, "GP0")

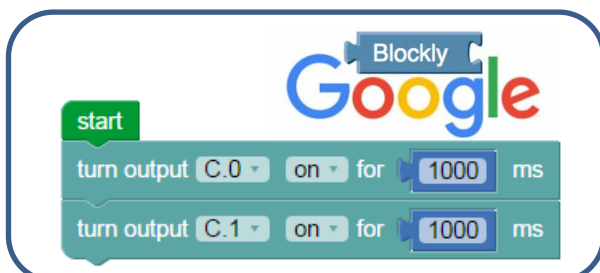
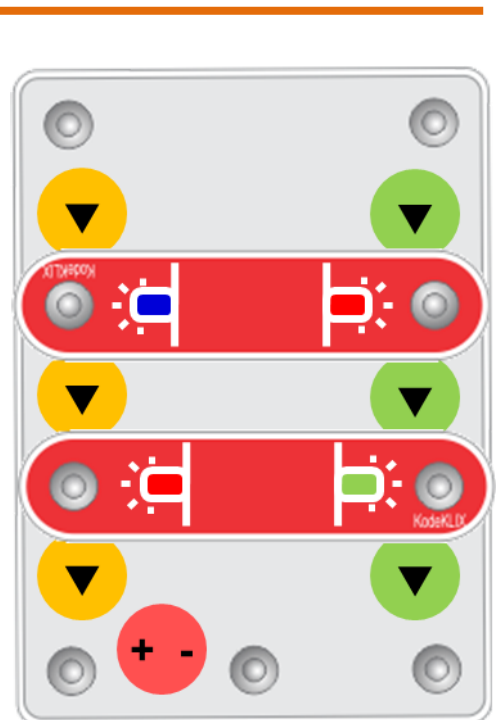
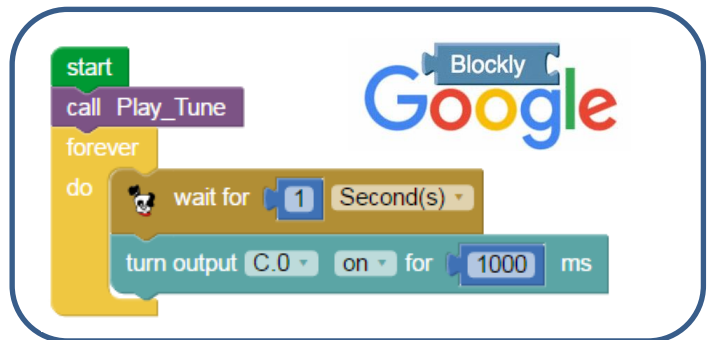
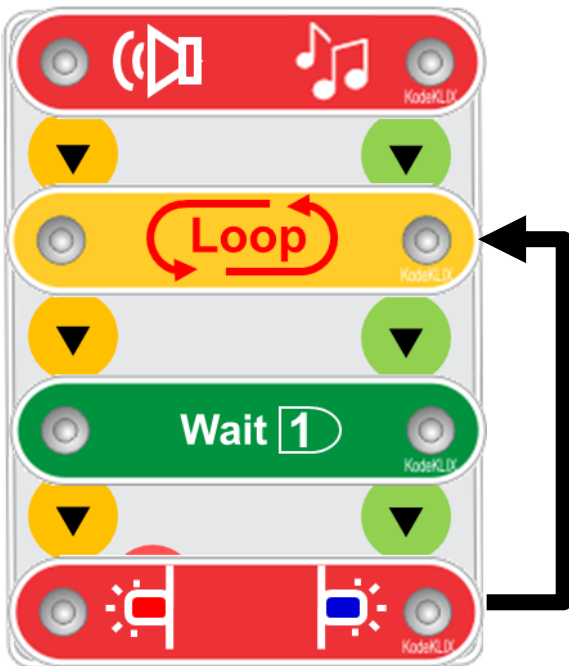
## ---- Code ---- ##
while True:
    time.sleep(1)      # wait for a second
    GP0.setPin(1)     # turn the LED on
    time.sleep(1)     # wait for a second
    GP0.setPin(0)     # turn the LED off
```



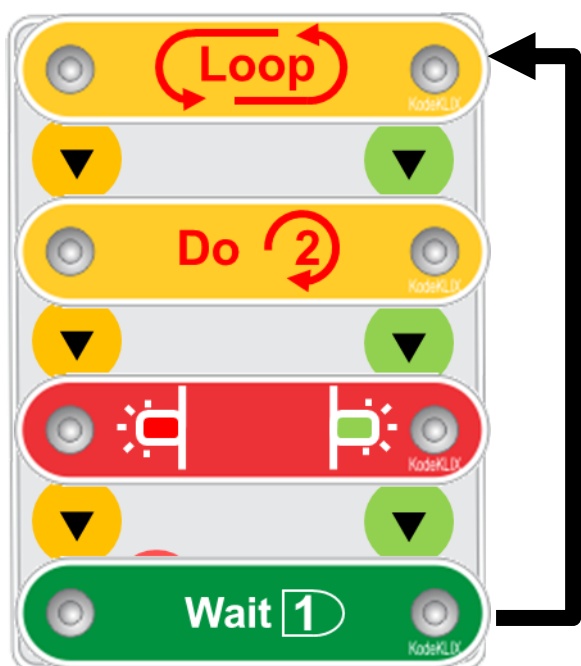
Coding Comparisons



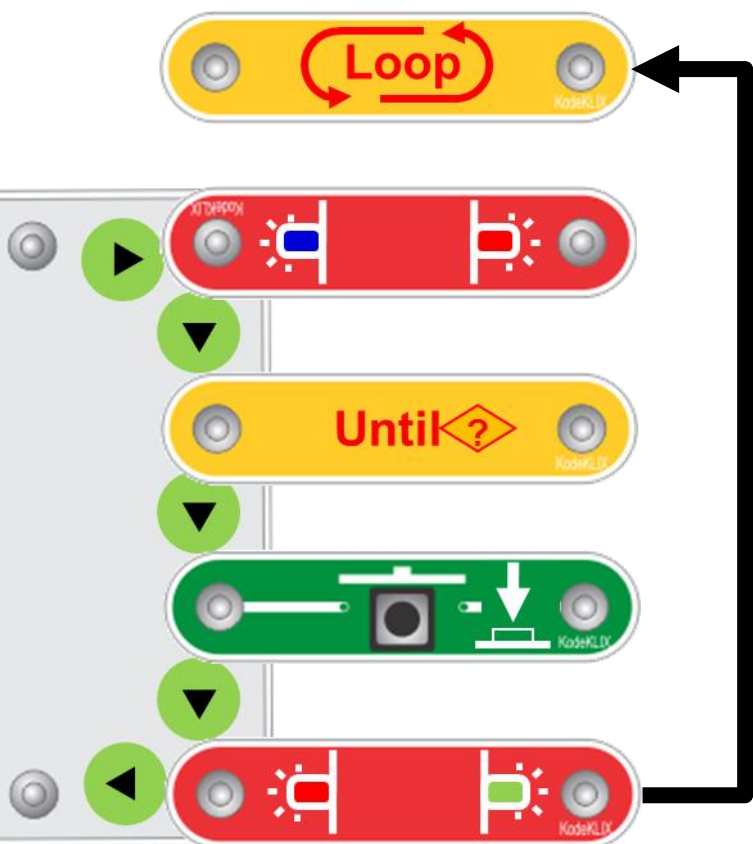
** note: the Play_Tune function is not shown to simplify the coding comparison. The tune function itself is quite complex.*



Coding Comparisons



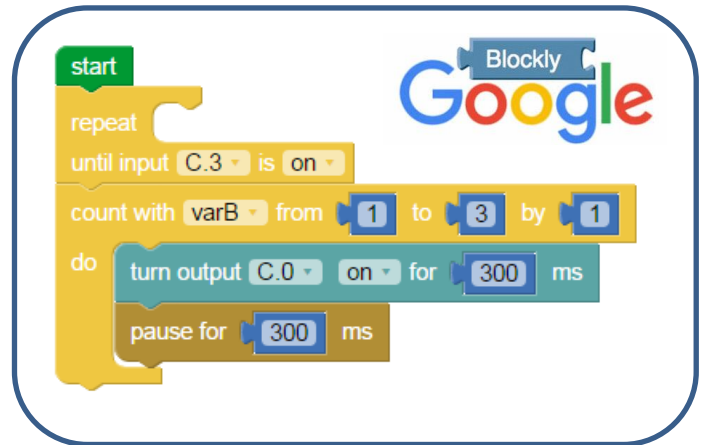
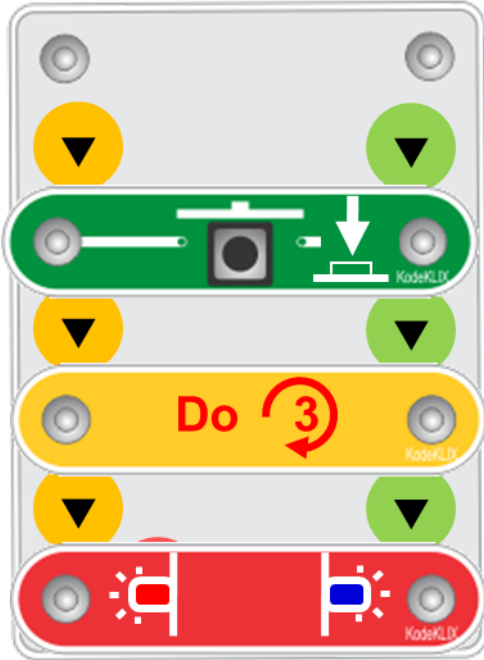
```
Blockly  
Google  
start  
forever  
do  
  count with varA from 1 to 2 by 1  
  do  
    turn output C.0 on for 500 ms  
    pause for 500 ms  
  wait for 1 Second(s)
```



```
Blockly  
Google  
start  
forever  
do  
  repeat  
    turn output C.0 on for 1000 ms  
  until input C.3 is on  
  turn output C.1 on for 1000 ms
```



Coding Comparisons



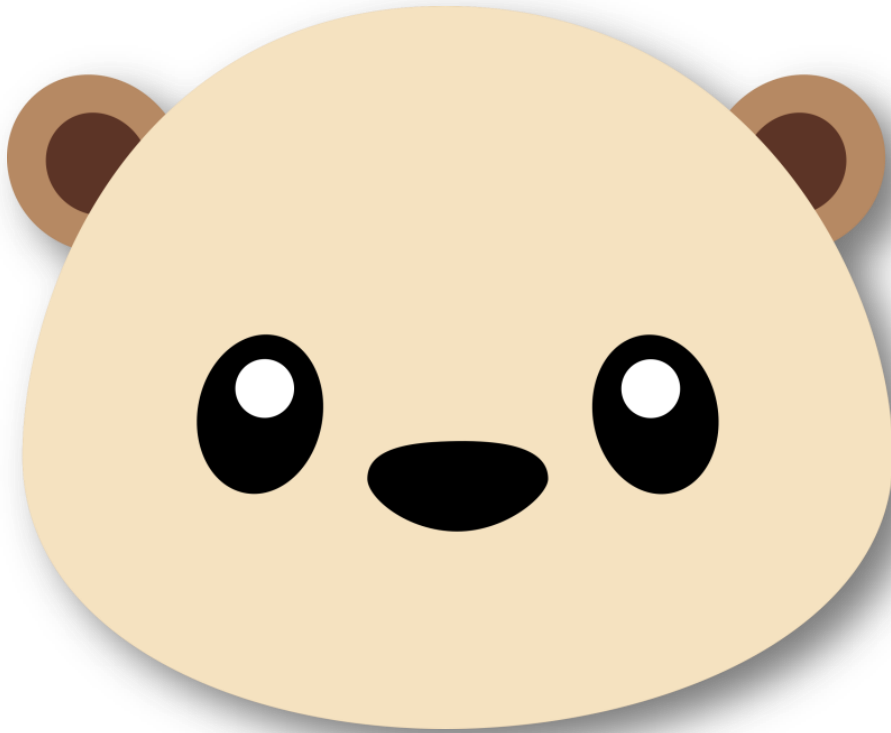
*Large print format of the **Coding with Oscar** Idea Book series follows.*

This format is ideally suited to in-class activities with larger groups. Parents can download and print the standard Ideas Booklets in full-colour from:

www.codingwithoscar.com.au

Printing rights are approved only for personal and non-commercial use.

Ideas Book One



Oscar gets a present
Oscar builds a torch
Up in the night sky
A fire engine in the forest
Oscar goes to the city
Oscar starts school

Ideas Book Two



Oscar is late for school
A night light for Possum
A gift for Oscar
A “thank you” for Possum
Possum keeps her stuff safe

Large format print
plus
Parent/Teacher
answers

Ideas Book Three



Oscar's other friend
Oscar's toy light sabre
Playing games
Lights for Christmas
Playing more tunes

Large format print
plus
Parent/Teacher
answers



Want more stories?

Do you enjoy the adventures of Oscar and his friends Possum, Baxter and others?

Want to see what they get up to next?

How about you think about some adventures they can have and see if you can create some gadgets and code to help them have fun and discover answers to their questions.

Did you know with four coding steps and 14 coding KLIX in your starter kit that there are over 25,000 coding possibilities! Have a play and see what fun and useful things you can create.

Decoration and construction ideas

Many of the projects and coding stories could be even more fun if you decorate the KodeKLIX® code block? Trying making the stories come to life by decorating a cardboard box to put over your KodeKLIX. Maybe you could make your box from play bricks such as Lego, Duplo... maybe it has wheels, or maybe it has wings. Let's see what your imagination can come up with!