

# KodeKLIX for PUP

---

## Programming Overview



# Programming Overview

- Programs (Apps) are created using sequenced instructions specific for the computer hardware (gadget)
- The instructions form the language of the microchip controller
  - PICAXE's language is BASIC
  - The instructions control inputs and outputs to perform task(s) as designated by the App.

```
-----  
' Initialise TITLE SCREEN  
' * Top row title  
' * Bottom row Score and High  
-----  
{  
_DisplayTopLine:  
  gosub _ClearLCD  
  _byte1=$80  
  gosub _SendCmdByte  
  for b0=0 to 7  
    lookup b0, ("INVADER",03), _byte1  
    gosub _SendDataByte  
  next  
  
_DisplayBottomLine:  
' Player score display
```



# BASIC Language

#1

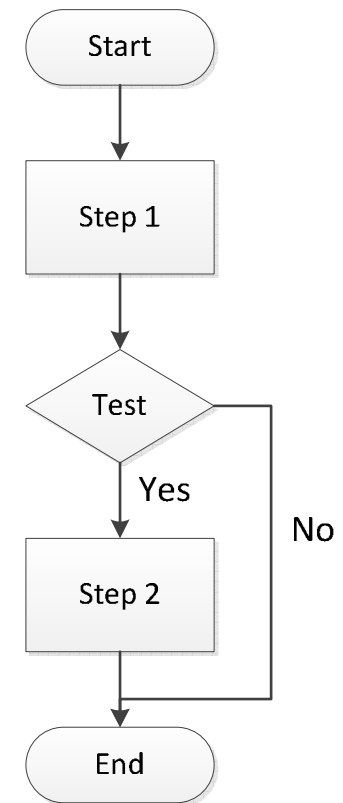
- BASIC actually stands for:
  - **B**eginners  
**A**ll-purpose  
**S**ymbolic  
**I**nstruction  
**C**ode
- It is an easy to understand, and easy to learn, coding language suitable for both beginners and advanced users
- PICAXE BASIC is a compacted version



# BASIC Language

#2

- BASIC is a written way to describe logic and operations such as those shown in a flow or process chart
- BASIC concepts
  - Variables
  - Statements
  - Conditional statements
  - Loops
- Detail can best be obtained from examples or the PICAXE guides





# BASIC - Variables

- Variables allow you to store and change information (ie numbers or values)
- Below are examples how to use variables:
  - $A = 10$  ' A is given a value of 10
  - $A = A + 1$  ' A now has value of 11
  - $LET A = A / 2$  ' A now 5, LET is optional
  - PICAXE variables are:
    - b0, b1, b2, etc which store values 0 to 255; or
    - w0, etc which store values 0 to 65535
    - w0 consists of b0 and b1



# BASIC - SYMBOLS

Start

- The SYMBOL statement lets you give sensible human names to variables
- Conventionally, SYMBOLS are placed at the start of your App's code
- For example:
  - SYMBOL A = b0 ' from now on use A
  - SYMBOL BYTE = A
    - ' you can have multiple
    - ' names for a variable



# BASIC - LABELS

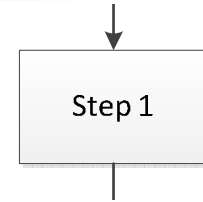
- Labels are named locations
- Program flow can branch to locations defined by labels
- Like symbols, all label names begin with an alphabetic character, end with a colon : and usually are placed at the beginning of the coding line, eg:
  - EndGame:
  - EnemyEnd:
  - FireRockets:

Start



# BASIC - Statements

- Statements
  - Include *commands* and *functions*
  - *Commands* tell the PICAXE to do something, sometimes with parameters
  - *Functions* return a response to a variable
- Full listing of *commands* and *functions* is available in the KodeKLIX helper or other PICAXE documentation and guides
- Examples in later sections will show you the most important statements for PUP







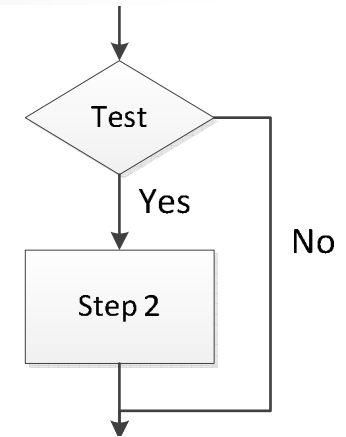
# BASIC - Conditional Statement

- Conditional statements

- At various points in your application's code you will find that you need to make decisions about inputs, data and results

- The basic decision framework is the IF...THEN...ELSE... structure

- Alternate,  
SELECT...  
CASE...  
ENDSELECT



```
IF variable ?? value {AND/OR variable ?? value ...} THEN  
{code}  
ELSEIF variable ?? value {AND/OR variable ?? value ...} THEN  
{code}  
ELSE  
{code}  
ENDIF
```



# BASIC - Loops

- Infinite loops – repeat forever

```
Label:  
(other program lines)  
GOTO Label
```

- Counting loops – repeat for a set number

```
FOR variable = start TO end {STEP {-}increment}  
(other program lines)  
NEXT {variable}
```

- Conditional loops – repeat...until...

```
DO  
{code}  
LOOP UNTIL/WHILE variable ?? COND
```

```
DO UNTIL/WHILE variable ?? COND AND/OR variable ?? COND...  
{code}  
LOOP
```



# Programming Input/Output

- *Commands / functions* used to control I/O
- Inputs includes:
  - User interface (Buttons, ThumbPad, etc)
  - Electrical signals, such communications from other devices
  - Variables / data
- Outputs include:
  - Data to LCD screen
  - Beeps, squeaks, and other sounds



# BASIC – Syntax Checking

- Syntax checking (keyword & labels only) can be performed as you enter the code on the keyboard (if selected)
- Different colours are used to identify statements, variables, constants, labels, etc
- Anything not identified defaults to black; check these uncoloured words for typing accuracy



```
-----  
' Initialise TITLE SCREEN  
' * Top row title  
' * Bottom row Score and High  
-----  
{  
_DisplayTopLine:  
  gosub _ClearLCD  
  _byte1=$80  
  gosub _SendCmdByte  
  for b0=0 to 7  
    lookup b0, ("INVADER",03), _byte1  
    gosub _SendDataByte  
  next  
  
_DisplayBottomLine:  
' Player score display  
-----
```



# Programming Structure

- To simplify programming, KodeKLIX imposes a structure which is helpful to beginners, but still flexible for experienced users
- Programs, just like stories, have:
  - *Beginning* to initialise conditions during start-up for the desired outcome;
  - *Middle* where conditions are monitored and responded to;
  - *End* to gracefully finish...



# KodeKLIX Default Structure

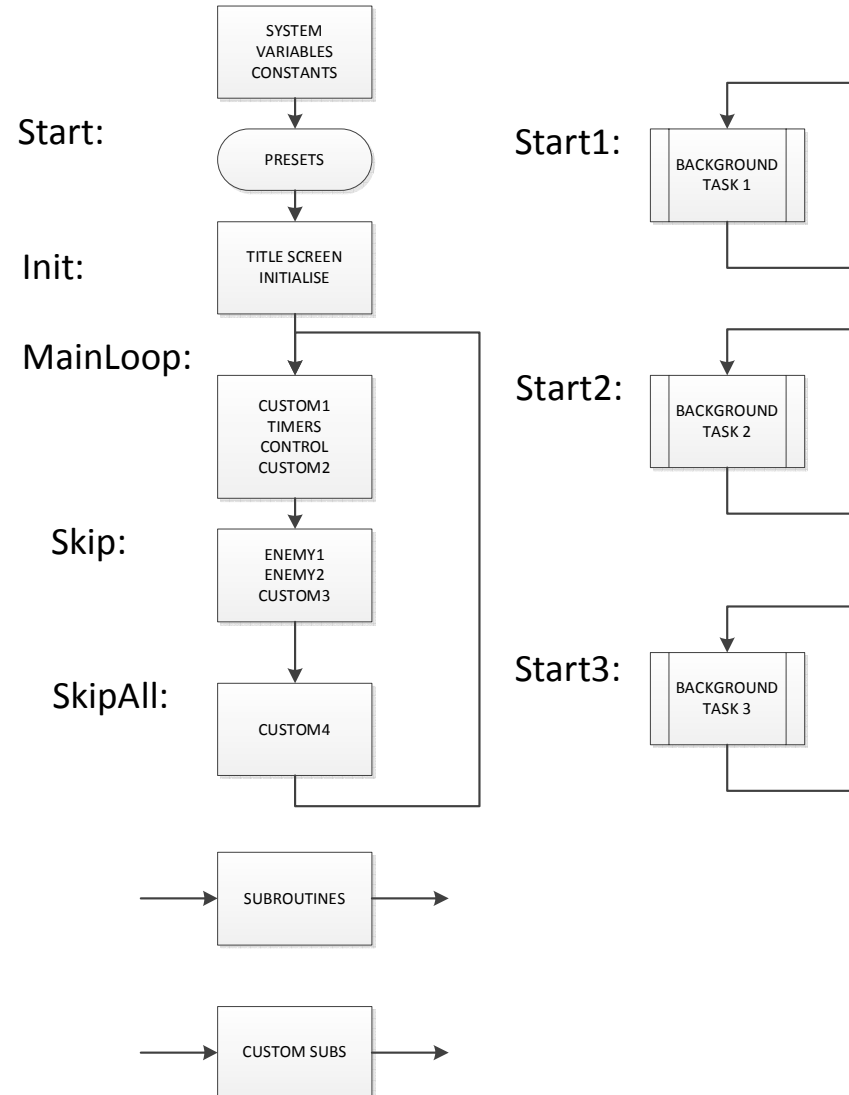
Setup Code  
(runs firsts)

Mainloop  
Code  
(repeats)

Subroutine  
Code  
(called)

The screenshot shows the following configuration options:

- System:** Presets (checked), LCD (checked), Title Screen (checked)
- Init:** Initialise (checked)
- MainLoop:** Custom 1 (unchecked), Timers (checked), Controls (checked), Custom 2 (unchecked)
- Skip:** Enemy 1 (checked), Enemy 2 (checked), Custom 3 (unchecked)
- SkipAll:** Custom 4 (checked), goto MainLoop
- Update:** Update LCD (checked), Update Scores (checked), Update Lives (checked), Update Levels (checked), Custom Subs (checked), PlayerMap (unchecked)
- Background tasks:** Start1: Task 1 (goto Start1), Start2: Task 2 (goto Start2), Start3: Task 3 (goto Start3)





# KodeKLIX – Game API

#1

- API = Application Programming Interface
- An API is a structure which predefines some common labels, functions and routines to help quick start your coding experience
- The Game API includes:
  - Pre-defined input and output names
  - Predefined program structure
  - Routines for controlling the LCD
  - Example code for common game needs



# KodeKLIX – Game API

#2

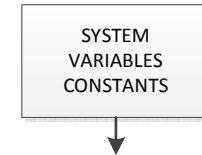
- SYMBOLs and labels used by the API are identified by an underscore in front of the name, eg `_MainLoop`, `_SendPosChar`
- Background tasks have predefined branch back coded in to the routines
- #macro system allows for customisation of the API and simplifies coding syntax for common functions, eg
  - `#print`, `#vprint`, `#cgXram`
  - Hand coding may be more efficient...





# KodeKLIX – Setup and Start

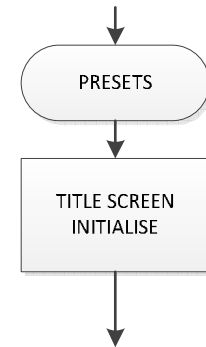
- Setup section of the code consists of three groupings
  - “System” definitions are provided by KodeKLIX to name resources of the chosen gadget, eg the PUP. They are read-only.
  - “Variables” definitions are editable, though there are several snippet options, eg SYMBOL port=b0
  - “Constants” definitions are similar to the Variables except they apply to numeric constants only, eg SYMBOL A=0





# KodeKLIX – Presets & Initialise

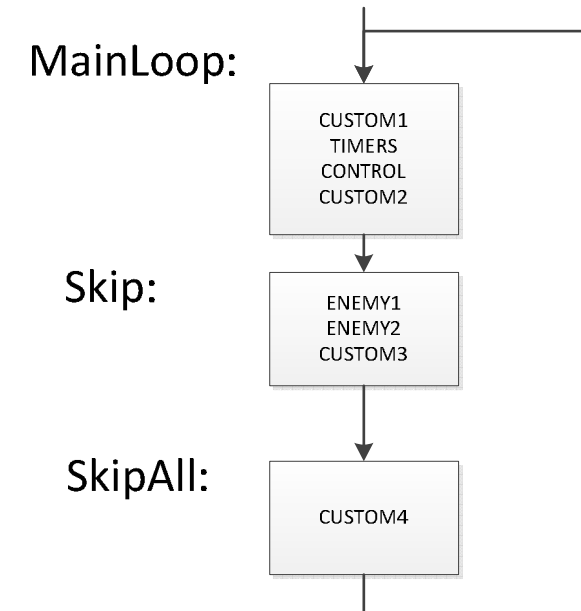
- Start0 is the start of “runnable” application code
  - “Presets” routine allows code which runs once (at `_PowerOnReset`) or may run multiple times, `_WarmReset`
- TitleScreen is optional, but most apps do...
- `_Init` is the start of application code run after the TitleScreen
  - This is where a new game is setup; may overlap with common new level coding





# KodeKLIX – the Mainloop

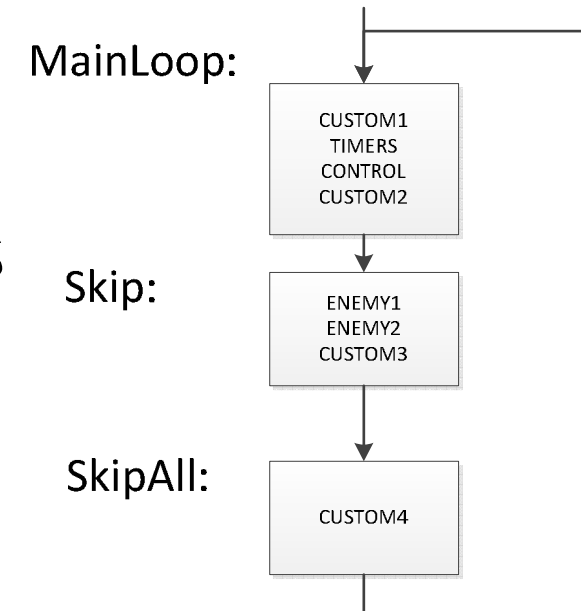
- `_Mainloop` is the work engine of the program. This loop continues until:
  - Power is switched off
  - Your code ends
- Within the `_Mainloop` are:
  - Timed events
  - Checking for user inputs
  - Typical player/enemy gaming routines
  - Custom routines





# KodeKLIX – *Skip Ahead Labels*

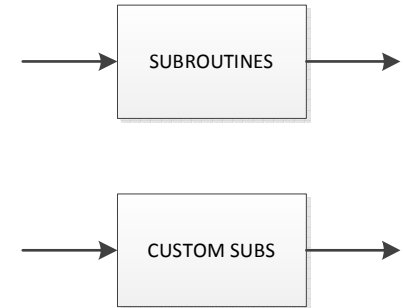
- Default App. structure allows for sequenced instructions and exclusions
- `_Skip` and `_SkipAll` labels are predefined
- Users can define their own labels, but these may not be interchangeable with another users code
- After the Custom4 coding routine `GOTO _MainLoop` occurs forming an endless loop





# KodeKLIX – Subroutines

- Subroutines are code modules which can be called from different parts of the main program stream
  - Usually started with *GOSUB*, end with a *RETURN*
    - *Player updates, Display updates, etc*
  - Some KodeKLIX subroutines can also end with a *GOTO* (or branch back)
    - *Level updates, Lives updates, etc*





# KodeKLIX – Background Tasks

- PICAXE 20M2 chip is capable of running multiple tasks, the standard API is sharing time between four *programs*
  - Start0 is the main routine
  - Start1 is configured for background music
  - Start2 is configured for fine timing
  - Start3 is configured as the randomiser
- Note: background should not stall the other routines with uninterruptable coding

