CatCore Hyper Datasheet

CatCore is a state of the art hardware coprocessor for managing your army of Cats. Being made by the top cat managers in the country, we guarantee the security and effectiveness of your control, managing your attack formations, data inputs and much more!

1. Modes

CatCore has a few different modes

Mode	Configuration Bits	Description
Button	`3'b000`	Interface with the buttons
UART	`3'b011`	Pass in Instructions here
Chall Secure Memory	`3'b010`	
Chall Fast Secure Memory	`3'b100`	

Set the related pins on the configuration pins to access that mode.

2. Legacy

To maintain backwards compatibility with CatCore, CatCore Hyper functions in CatCore Legacy Mode first. The new features are implemented using reserved opcodes. This allows CatCore Hyper to be used as a drop in replacement for any CatCore Products.

Instruction Set

The instruction format is usually 18 characters long (except for Kill Cats), as such

```
<opcode character><16 data characters><same opcode character>
```

Instruction	Opcode (ASCII Character)	Format
Kill Cats	A	This instruction is only 3 characters long
Read Data	@	You can read some keys from here
Executor	D	Runs catcore hyper Instructions

Instruction - Kill Cats

Format

```
A<selector character>A
```

Selector Characters

1. A -> Cat 0 Dies
2. B -> Cat 1 Dies
3. C -> Cat 2 Dies
4. D -> Cat 3 Dies
5. E -> Cat 4 Dies
6. F -> Cat 5 Dies
7. G -> Cat 6 Dies
8. H -> Cat 7 Dies
9. ` -> All Cats are Revived

Sample Instruction

```
ABA -> Cat 1 dies (is lit up)
```

Instruction - Read Data

Format

```
@---------------^@

@ opcode (Read Data)
- irrelevant character
^ Replace with selector character
```

Selector Characters

1. `A` Read Key for Hornet Revenge
2. `B` Read Flag for SMD Skills Oscillator Challenge - Only works if an oscillator is installed

Example instructions

```
@---------------B@     <- Reads Flag for SMD skills oscillator challenge
```

Instruction - Executor

Format

```
D<encrypted or devmode instruction>D
```

The hyper instruction is 16 characters long and the format is described later below

For example, an Instruction to display the flag (under devmode) is as such

```
D@---------DEV@D
```

3. Hyper

CatCore Hyper mode implements new features, such as a memory system which can store keys, and an advanced Cat LED control System.

However, to protect our users, we have decided to use our special XOR encryption to protect our hyper instructions. (instruction ^ key == program). When passing in the instruction into the executor, our hardware security XOR decrypts the instruction, and then runs it following the bottom instruction set.

Instruction Set

The instruction format is usually 16 characters long

```
<opcode character><14 data characters><same opcode character>
```

Instruction	Opcode (ASCII Character)	Format
Dev	@	Display Dev Welcome Message. Data Characters are discarded
Flag	A	Displays flag. Only done when the data characters == "1w4n7myfl49p15"
LED Control	B	Controls the LEDs and PWM brightness
Memory	C	Reads from memory and outputs the value through UART. You might need to send the instruction twice for it to work

Instruction	Opcode (ASCII Character)	Format
		In dev mode you have access to

Instruction - LED Control

This instruction gives you full control over your cats. Not just by killing them, but allows you to change their attack pattern (led on off), their attack rate (PWM) and much more.

Format

```
B<enable character><led control character><PWM Val 1><PWM Val 2>----------B
```

1. Enable Character
    1. A to turn on manual control of LEDs
    2. B to turn off
2. LED control Character
    1. 8 bit value of what LEDs to turn on. Each LED represents one but
3. PWM Val 1 and PWM Val 2
    1. The LED brightness is determined by PWM Val 1 / PWM Val 2
    2. PWM Val 1 is the PWM on cycles
    3. PWM Val 2 is the total number of PWM cycles

Instruction - Memory

This instruction is pipelined (run the same instruction 2 times and the last output is the correct one)

Format

```
C<address character>-------------C
```

The address character is a value ranging from 0 to 7 (3 bits) in developer mode.

The memory stores useful values such as XOR encryption keys that might be useful for other programs

Developer Mode

To enable developer mode, make sure you set pin N8 of the Lattice Chip to GND on startup. This give you 2 main benefits

1. You can run instructions without signing, provided that your last 3 data characters are `DEV`
2. Pins are exposed for debugging + button presses reveal debugging modes on the LEDs

Example of Dev mode instruction - this prints the developers message

```
@---------DEV@   The DEV signature must be there (and the badge must be in dev mode) to run
```

Pinout

1. `A10 -> memAddress[0]`
2. `A9 -> memAddress[1]`
3. `A8 -> memAddress[2]`
4. `A3 -> memAddress[3]`

4. Appendix

If there are any errors or confusion in the datasheet, feel free to contact Hackin7 (or any staff nearby).

https://github.com/Hackin7