Super Nintendo Entertainment System (SNES) Controller

Jalal Kawash

Outline

1. SNES controller

Section 1

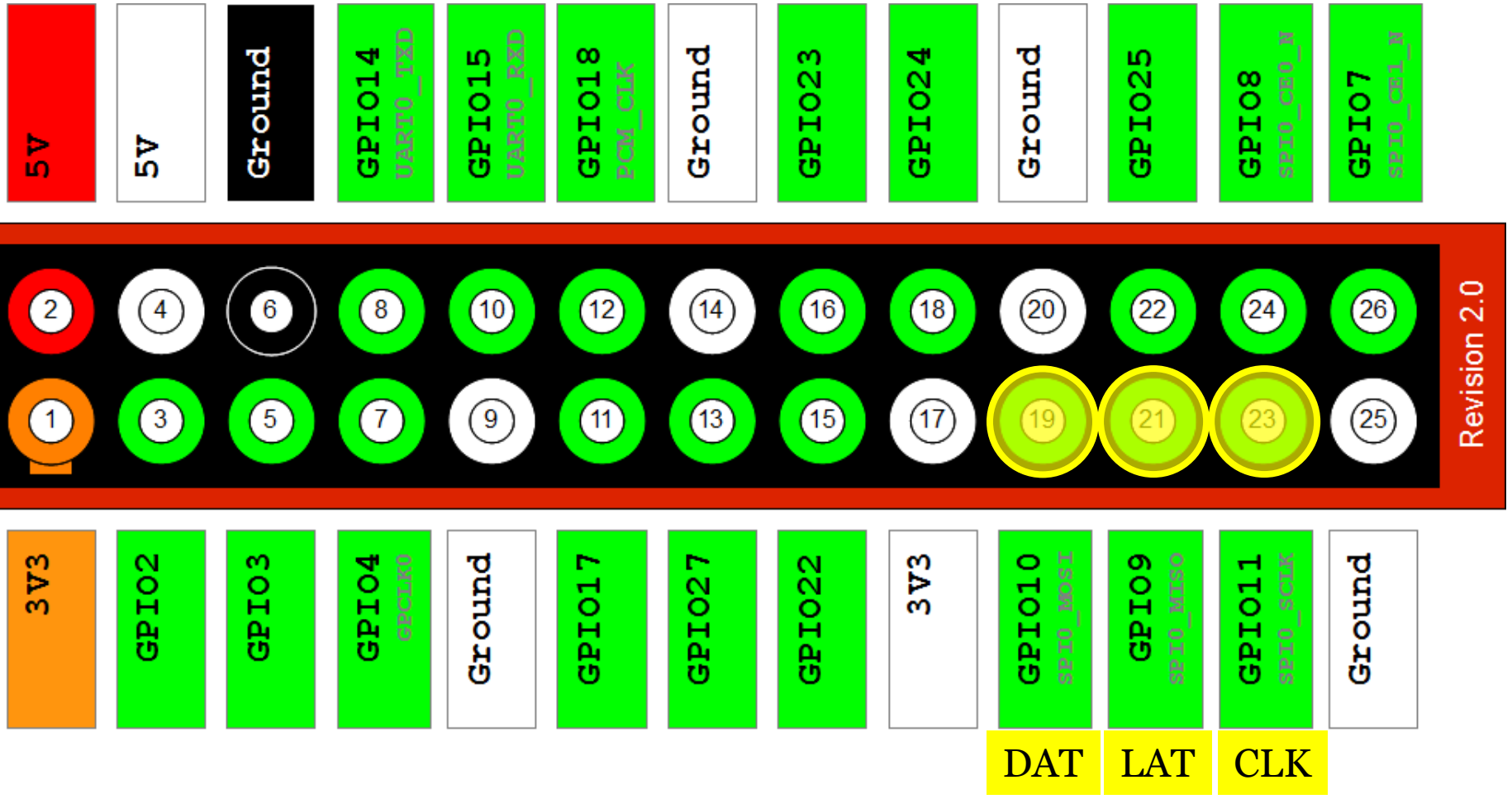
SNES Controller

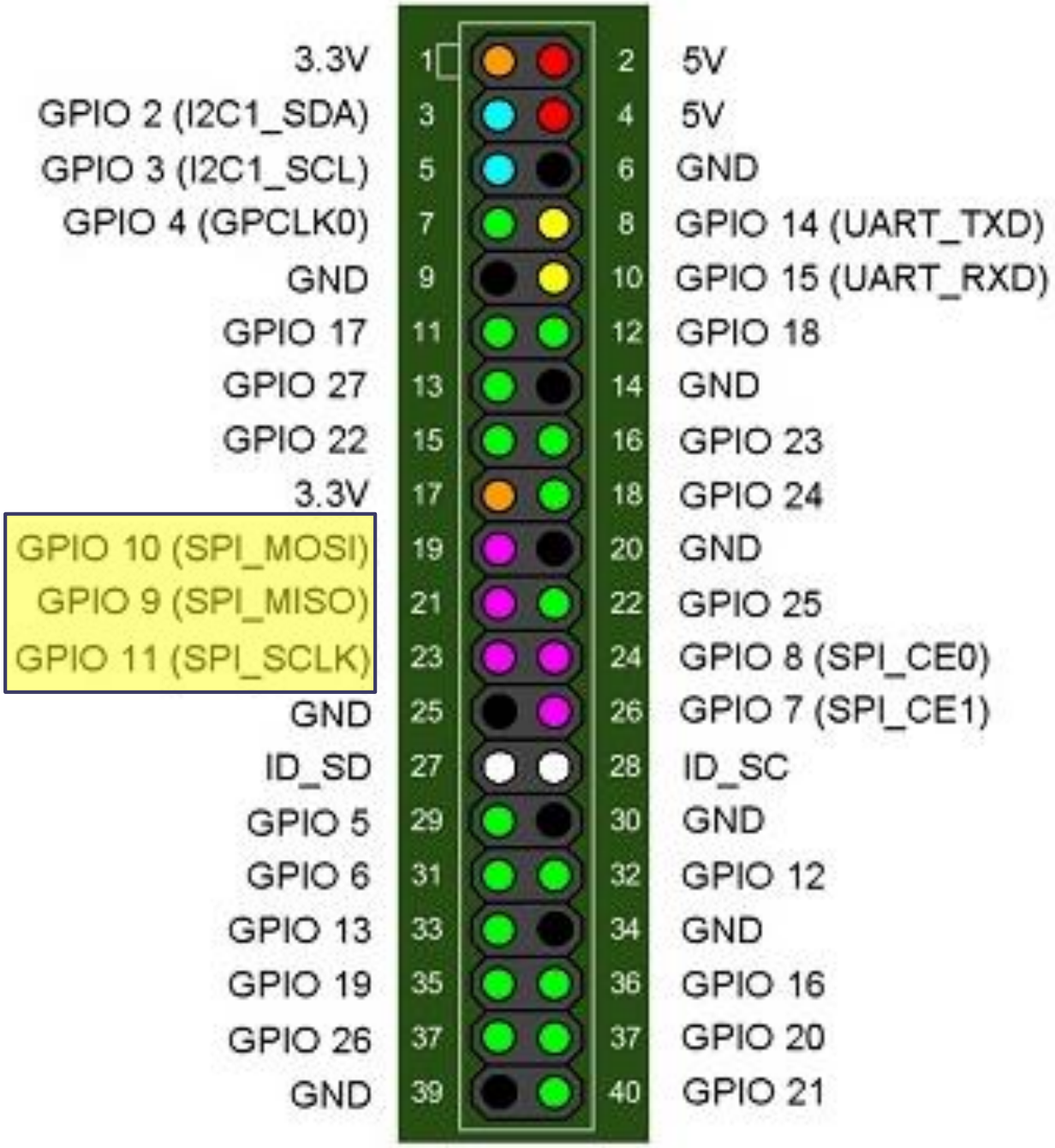
Section 1 Objectives

At the end of this section you will

1. Understand the communication protocol of the SNES controller
2. Write programs to interact with the SNES controller

GPIO SNES Pins




3.3V 1
2 5V
GPIO 2 (I2C1_SDA) 3
4 5V
GPIO 3 (I2C1_SCL) 5
6 GND
GPIO 4 (GPCLK0) 7
8 GPIO 14 (UART_TXD)
GND 9
10 GPIO 15 (UART_RXD)
GPIO 17 11
12 GPIO 18
GPIO 27 13
14 GND
GPIO 22 15
16 GPIO 23
3.3V 17
18 GPIO 24
GPIO 10 (SPI_MOSI) 19
20 GND
GPIO 9 (SPI_MISO) 21
22 GPIO 25
GPIO 11 (SPI_SCLK) 23
24 GPIO 8 (SPI_CE0)
GND 25
26 GPIO 7 (SPI_CE1)
ID_SD 27
28 ID_SC
GPIO 5 29
30 GND
GPIO 6 31
32 GPIO 12
GPIO 13 33
34 GND
GPIO 19 35
36 GPIO 16
GPIO 26 37
37 GPIO 20
GND 39
40 GPIO 21

The Controller

SNES Button Numbers

Button Number/ Clock Cycle	Button
1	B
2	Y
3	Select
4	Start
5	Joy-pad UP
6	Joy-pad DOWN
7	Joy-pad LEFT
8	Joy-pad RIGHT
9	A
10	X
11	Left
12	Right
13-16	Unused – for future use (always high)

SNES Lines

- LATCH: latch line 
  - Used by the core to instruct the controller to latch the state of the buttons internally
    - Which buttons on the controller are pressed
- CLOCK: clock line
  - The core sends a clock signal of 16 intervals 
    - One for each button
    - Interval 1 is for Button B; 2 for Y etc …
- DATA: data line
  - Controller sends serial signal to core indicating which buttons are pressed 

Recall GPFSEL{n}

GPIOFSEL{n}

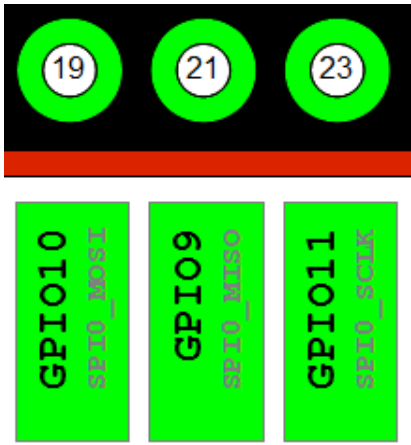
30-31	27-29	24-26	21-23	18-20	15-17	12-14	9-11	6-8	3-5	0-2
reserved	Pin {n}9	Pin {n}8	Pin {n}7	Pin {n}6	Pin {n}5	Pin {n}4	Pin {n}3	Pin {n}2	Pin {n}1	Pin {n}0

Initializing SNES – CLOCK line

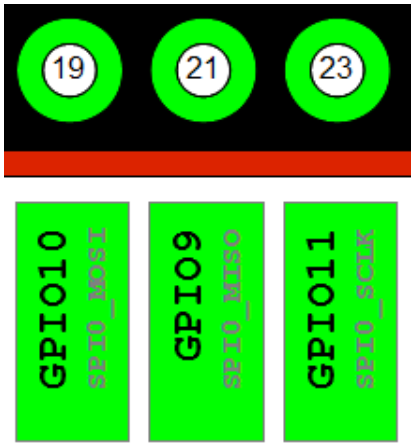
- Set GPIO pin 11 (CLK) to output

```
#define      CLK          11
#define      LAT          9
#define      DAT          10


INP_GPIO( CLK );
```



Initializing SNES



- Similarly,
- Set GPIO pin 9 (LATCH) to output
- Set GPIO pin 10 (DATA) to input (code 0)

```
INP_GPIO( LAT );
OUT_GPIO( LAT );
INP_GPIO( DAT );
```

```
#define GPIO_BASE   0x3F200000

static unsigned *gpio = (unsigned*)GPIO_BASE;          // GPIO base

// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x) or SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |= (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |= (((a)<=3?(a)+4:(a)==4?3:2)<<(((g)%10)*3))
```

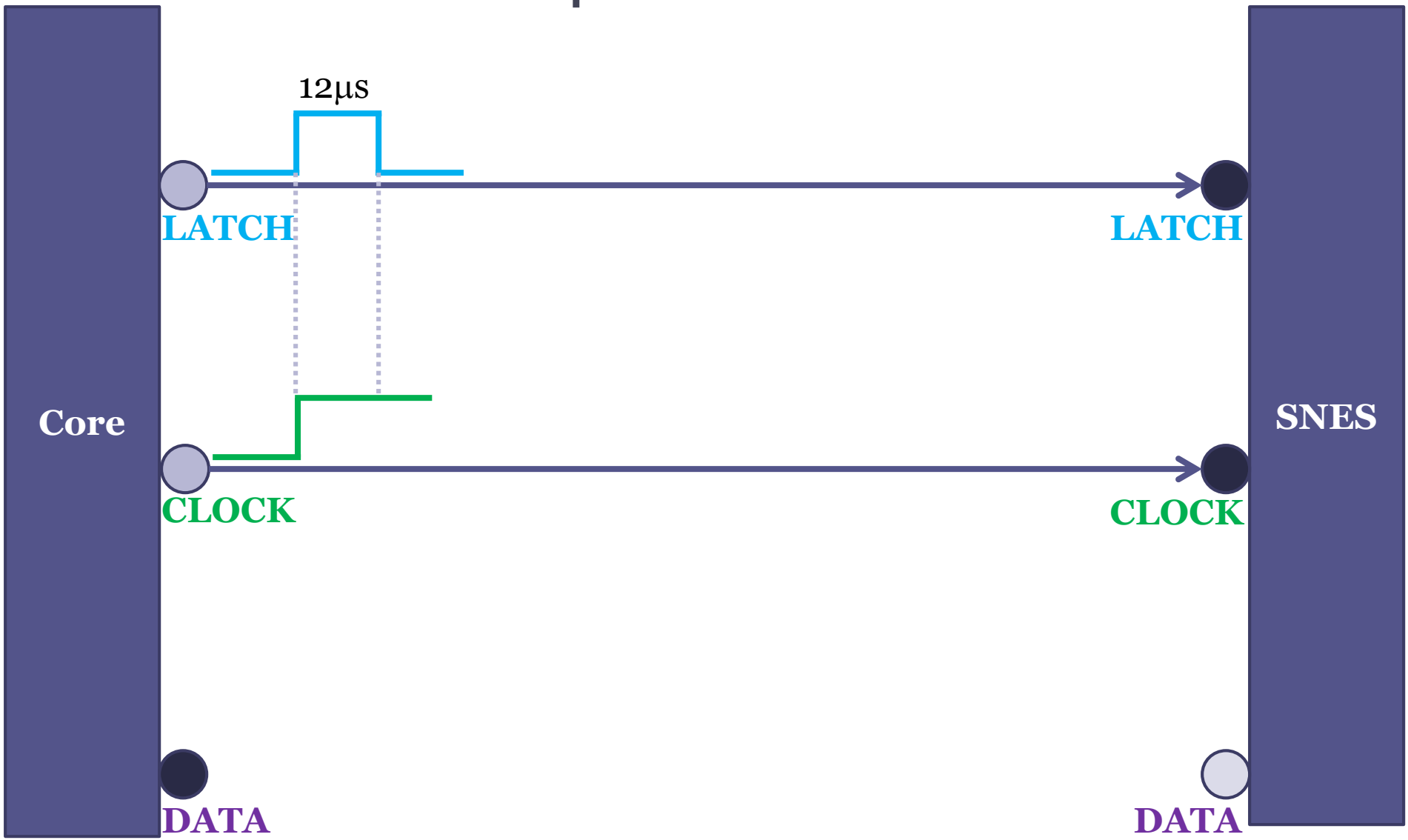
```
#define     CLK         11
#define     LAT         9
#define     DAT         10


void initSNES() {
    INP_GPIO( CLK );        // CLK
    OUT_GPIO( CLK );
    INP_GPIO( LAT );        // LATCH
    OUT_GPIO( LAT );
    INP_GPIO( DAT );        // DATA
}
```

Communication Protocol

- LATCH is used to instruct the SNES to sample and latch buttons
  - Which buttons are pressed
  - Latched in a register
- CLOCK line is used to synchronize serial signal sent from SNES to the core
- DATA line is used to read the serialized latched register representing button sampling

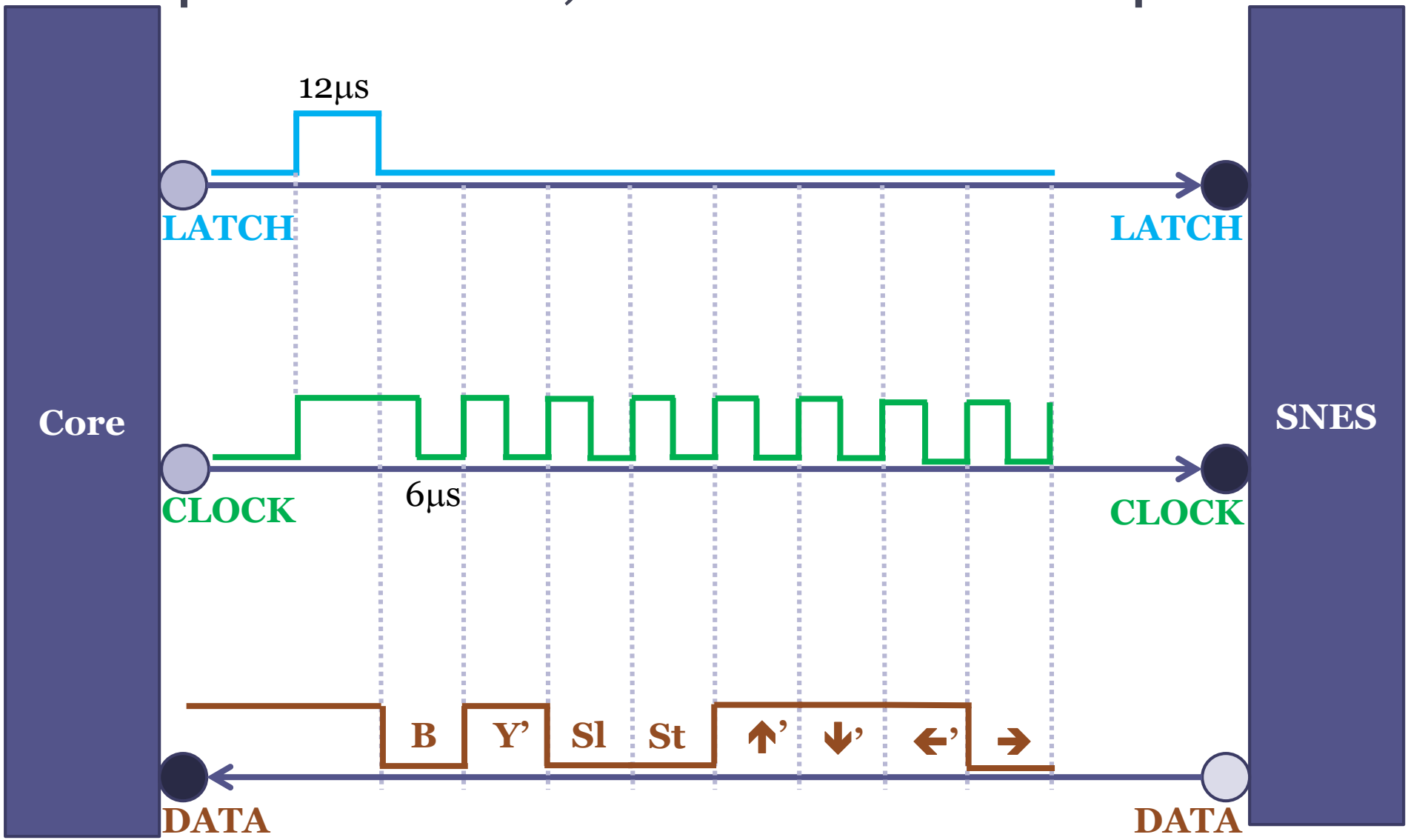
Communication Protocol (1)
Core Asks for a report on buttons



Communication Protocol (2)
SNES Samples Buttons

- Sampling buttons = creating a register representation of which buttons are pressed
- One bit for each button is sufficient
  - Example: 1111101111100
  - Three buttons are pressed (buttons 0, 1, & 7)

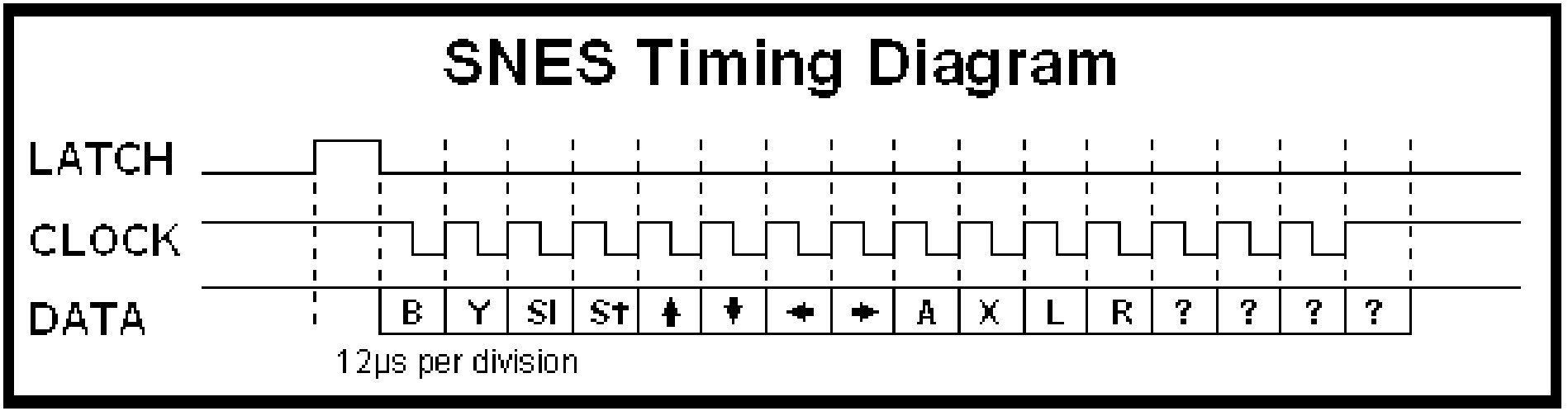
Communication Protocol (3)
Core pulses clock; SNES to sends report



Communication Protocol (3)
Core pulses clock; SNES to sends report

- Pulsing continues for 16 times
- SNES has 12 buttons
  - First 12 pulses sample the buttons
- Last 4 pulses are not used (always 1)
  - Reserved for future extensions

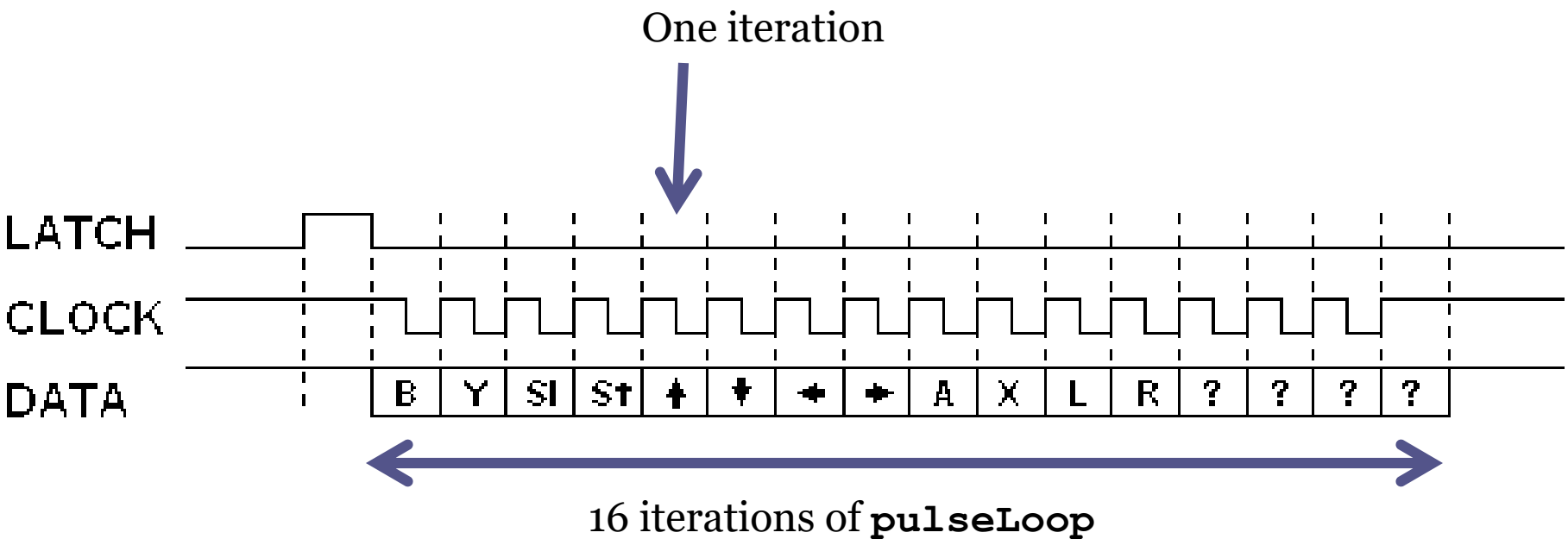
Communication Protocol (3)



From: homepages.inf.ed.ac.uk

Reading from SNES

1. MOV buttons, #0 // register sampling buttons
2. writeGPIO(CLOCK,#1)
3. writeGPIO(LATCH,#1)
4. wait(12μs) // signal to SNES to sample buttons
5. writeGPIO(LATCH,#0)
6. pulseLoop: // start pulsing to read from SNES
    1. i = 0
    2. wait(6μs)
    3. writeGPIO(CLOCK,#0) // falling edge
    4. wait(6μs)
    5. readGPIO(DATA, b) // read bit i
    6. buttons[i] = b
    7. writeGPIO(CLOCK,#1) // rising edge; new cycle
    8. i++ // next button
    9. if (i < 16) branch pulseLoop


One iteration
LATCH
CLOCK
DATA
B
Y
SL
St
A
X
L
R
?
?
?
?
16 iterations of pulseLoop

System Timer

CLO Register	
Synopsis	System Timer Counter Lower bits. The system timer free-running counter lower register is a read-only register that returns the current value of the lower 32-bits of the free running counter.

Bit(s)	Field Name	Description	Type	Reset
31:0	CNT	Lower 32-bits of the free running counter value.	RW	0x0

- Address: 0x3F003004
- CHI is the next register, containing the higher 32-bits

Waiting

```
#define CLO_REG 0x3F003004

unsigned *clo = (unsigned*)CLO_REG;
unsigned c = *clo + 12; \\ micro Secs

while (c > *clo);
```