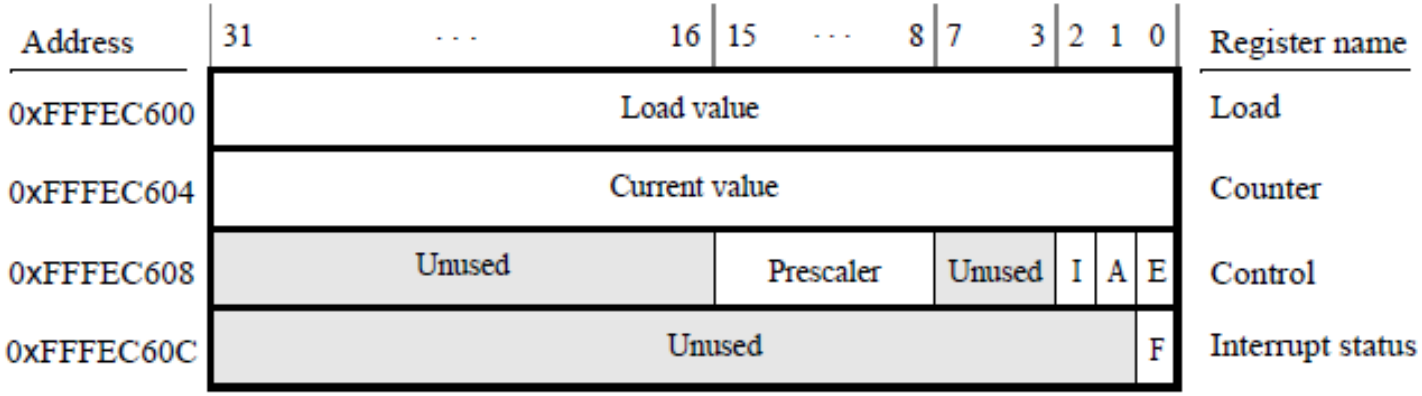
# Timer Example

2024

# ARM A9 MPCore Timers

- Base address 0xFFFEC600
- It can be read or written using word accesses.
- Clock frequency = 200 MHz




How does it work?

Timer is a down-counter. It starts counting from the value on the “Load” register. This is a control register for the timer.

When the timer is enabled, with each tick (1/200Million) the value is decremented by one. The current value is a data register for the timer, and it shows the counter.

# Load a value, enable the timer and watch the current value

Try it yourself:

- R0: Set a pointer for the timer base address 0xFFFEC600
- R1: R1 keeps the value that you want the timer start from.
- Store R1 to the “load register of the timer”.
- Now, prepare a control word on R1. You want Enable (E) to be ‘1’ and the rest to be ‘0’.
- Store R1 (the control word) to the “control register of the timer”.

# Solution

- Video: https://youtu.be/tkLw-14ibpw
- Code:

```
.global _start
_start:

LDR R0,=0xFFFEC600
LDR R1,=200000000
STR R1, [R0]
MOV R1, #0b11
STR R1, [R0, #8]

end: b end

.end
```

# Load a value, enable the timer and autoload and watch the current value

- Set (E) and (A) bits to ‘1’ and watch the result.

# Solution

- Video: https://youtu.be/tkLw-14ibpw
- Code:

```
.global _start
_start:

LDR R0,=0xFFFEC600
LDR R1,=200000000
STR R1, [R0]
MOV R1, #0b11
STR R1, [R0, #8]

end: b end

.end
```

# Task: Write “0xAAAAAAAA” to R5 1 second after the program is started.

- Configure the timer, enable it.
- Check the “F” bit on the interrupt **status** register. If it is ‘1’, it means one second is passed.
- Write “0xAAAAAAAA” to R5 if F is 1.

# Solution

- Video: https://youtu.be/tkLw-14ibpw
- Code:

```
.global _start
_start:

LDR R0,=0xFFFEC600
LDR R1,=200000000
STR R1, [R0]
MOV R1, #0b11
STR R1, [R0, #8]
STR R1, [R0, #0xc]  // reset the flag.

// check F bit:
loop: LDR R1, [R0, #0xc]
ANDS R1, #1
// when F is 1.
LDRNE R5, =0xAAAAAAAA
BEQ loop

end: b end

.end
```

# Task: Turn on all the LEDs 1 second after the program is started.

- Configure the timer, enable it.
- R2 points to the base register of the LEDs. (Address: 0xFF200000)
- R3 is the word to be written to the LEDs. Least significant 10 bits are 1, the others are not important.
- Check the “F” bit on the interrupt **status** register. If it is ‘1’, it means one second is passed.
- Write the content of R3 to LEDs base address (pointed by R2)

# Solution

- Video: https://youtu.be/tkLw-14ibpw
- Code:

```
.global _start
_start:

LDR R0,=0xFFFEC600
LDR R1,=200000000
STR R1, [R0]
MOV R1, #0b11
STR R1, [R0, #8]
STR R1, [R0, #0xc]  // this is here to be sure the flag is down
at the beginning.

LDR R2, =0xFF200000  // led pointer
LDR R3, =0b1111111111 // led pattern

// check F bit:
loop: LDR R1, [R0, #0xc]
ANDS R1, #1
// when F is 1.
STRNE R3, [R2]
BEQ loop

end: b end

.end
```

# Task: Flip all the LEDs with one second period.

- Configure the timer, enable it.
- R2 points to the base register of the LEDs. (Address: 0xFF200000)
- R3 is the word to be written to the LEDs. Least significant 10 bits are 1, the others are not important.

Repeat the rest forever:

- Check the “F” bit on the interrupt **status** register. If it is ‘1’, it means one second is passed.
- Write the content of R3 to LEDs base address (pointed by R2).
- Bitwise flip the least significant 10 bits of R3. You may flip more bits, whatever easier for you. You can use EOR.

# Solution

- Video: https://youtu.be/w9S3cnZtRp4
- Code:

```
.global _start
_start:

LDR R0,=0xFFFEC600
LDR R1,=200000000
STR R1, [R0]
MOV R1, #0b11
STR R1, [R0, #8]

LDR R2, =0xFF200000  // led pointer
LDR R3, =0b1111111111 // led pattern
LDR R4, =0b1111111111

// check F bit:
loop: LDR R1, [R0, #0xc]
ANDS R1, #1
// when F is 1.
STRNE R3, [R2]
EORNE R3, R3, R4
STRNE R1, [R0, #0xc]  // reset the flag.
b loop

end: b end

.end
```