MIPS based MCU Architecture

Final Project Assignment Definition

Hanan Ribo

08.07.25

Table of contents

1. Aim of the project

- Design, synthesis, and analysis of a simple (single cycle architecture) MIPS CPU core with Memory Mapped I/O, interrupt capability, and ***Serial communication peripheral (entitles to a 20% bonus)***
- ***Pipelined MIPS CPU core*** instead of a single-cycle core ***(entitles to a 10% bonus)***
- Understanding of CPU vs. MCU concepts, and FPGA embedded memory structures

2. Definition and prior knowledge

The aim of this project is to design CPU MIPS based MCU. The CPU will use a Single Cycle MIPS architecture and must be capable of performing full instruction set of simple MIPS (given as an appendix). The design will be located on Altera Board. The MIPS architecture is Harvard architecture in order to increase throughput and simplify the logic. For additional information regarding MIPS CPU, Architecture, ISA and instructions see MIPS technical documentation [1].

3. Assignment definition

The architecture must include a MIPS ISA compatible CPU with data memory DTCM and program memory ITCM for hosting the program data and code segments. The block diagram of an architecture is given in Figure 1. The CPU will have a standard MIPS register file. The top level and the MIPS core must be structural. The design must be compiled and loaded to the Altera board for testing. A single clock (CLK) should be used in the design.

Note: use push-button **KEY0 as a System RESET** (brings the PC to the first program instruction)

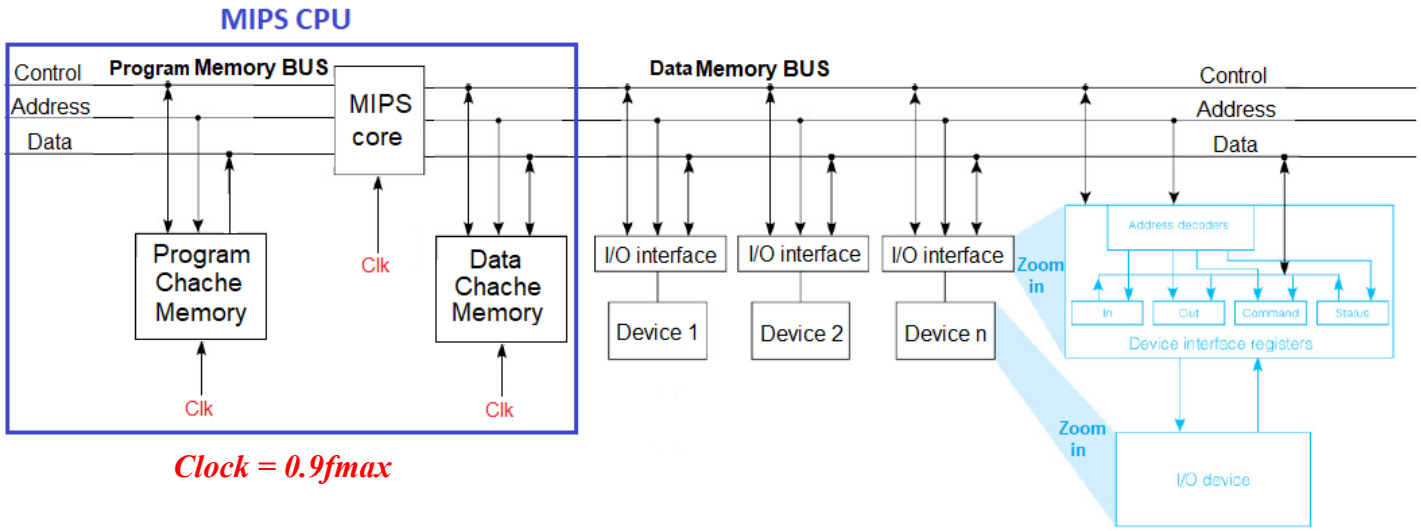
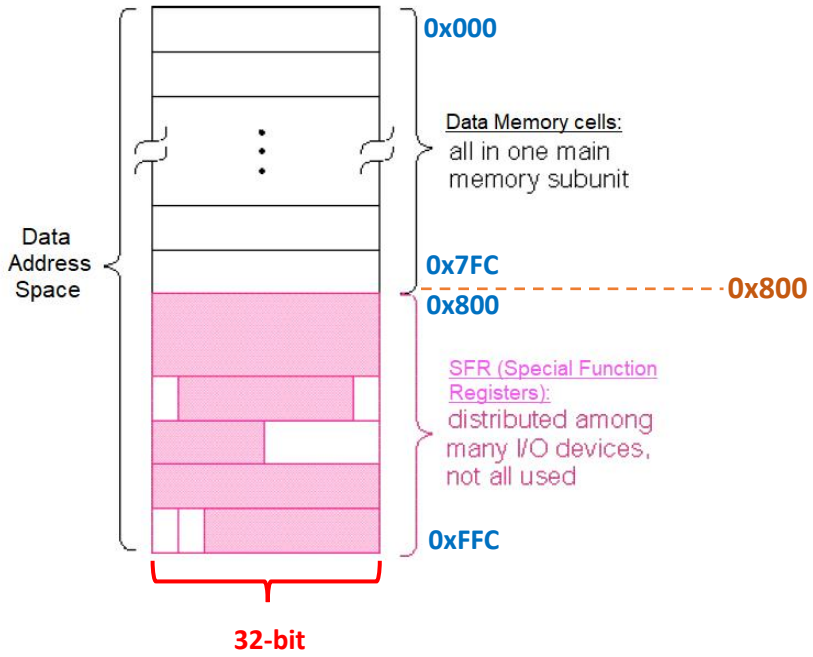


Figure 1 : MCU System architecture

- The GPIO (General Purpose I/O) is a simple decoder with buffer registers mapped to data address (Higher than data memory) as given in the assembly code examples that enables the CPU to output data to GPIO devices as LEDs and 7-Segment and to read the Switches array value.



The Data Address Space is 32-bit WORD aligned where the physical address space it is the lowest 12-bit $\mathbf{0}$ ... $\mathbf{0}\boldsymbol{A_{11}}$ ... $\boldsymbol{A_0}$ with partial mapping.

Figure 2: Data Address Space contains Data Memory and Memory Mapped I/O

I/O devices connected:

In the hardware test case, you will have to test an **ALU digital system** onto D10-Standard FPGA board.

- Board ***ten*** switches (SW9-SW0) and push ***four*** debounced pushbuttons (KEY3-KEY0) will be used as ***Input interface***.
- Board ***10*** red LEDs (LEDR9-LEDR0) and ***six*** 7-segment displays (HEX5-HEX0) used as ***Output interface***.
- Connections between the 2x20 GPIO Expansion Header and Cyclone V SoC FPGA

Figure 2a: I/O interface of the DE10-Standard FPGA board

Figure 2b: I/O interface of the DE2-115 FPGA board

```
#-------------------------------------------------------
#            MEMORY Mapped I/O
#-------------------------------------------------------
#define PORT_LEDR[7-0] 0x800 - LSB byte (Output Mode)
#-------------------- PORT_HEX0_HEX1 -------------------
#define PORT_HEX0[7-0] 0x804 - LSB byte (Output Mode)
#define PORT_HEX1[7-0] 0x805 - LSB byte (Output Mode)
#-------------------- PORT_HEX2_HEX3 -------------------
#define PORT_HEX2[7-0] 0x808 - LSB byte (Output Mode)
#define PORT_HEX3[7-0] 0x809 - LSB byte (Output Mode)
#-------------------- PORT_HEX4_HEX5 -------------------
#define PORT_HEX4[7-0] 0x80C - LSB byte (Output Mode)
#define PORT_HEX5[7-0] 0x80D - LSB byte (Output Mode)
#-------------------------------------------------------
#define PORT_SW[7-0]   0x810 - LSB byte (Input Mode)
#-------------------------------------------------------
```

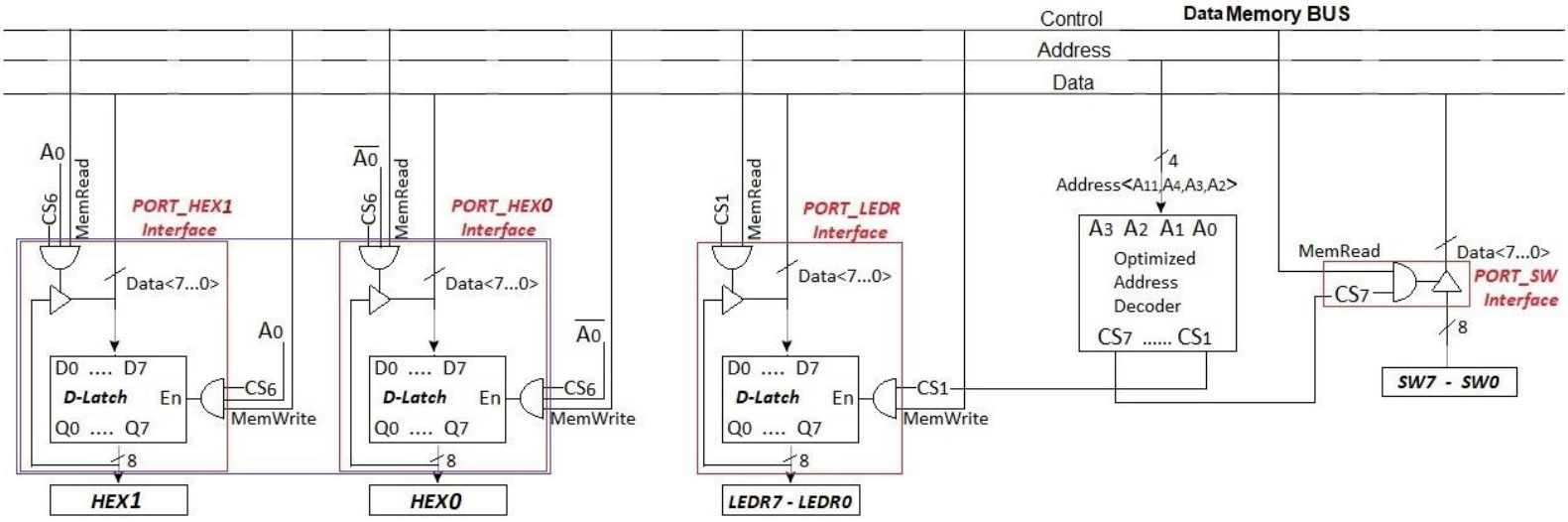


Figure 3: Primitive GPIO peripheral connection using Memory Mapped I/O approach

- The CPU will be based on the *standard 32bit MIPS ISA* and the Instructions will be 32 bit wide. The following table shows the MIPS instruction format. For more information, see MIPS technical documents [1].

Type	-31- format (bits) -0-					
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

Table 1 : MIPS Instruction format

The Data address space is 4kB. Memory latency will be according to Table 2

Memory	Write Latency	Read Latency
Program Memory (I-Cache / ITCM)	1 clk	1 clk
Data Memory (D-Cache / DTCM)	1 clk	1 clk

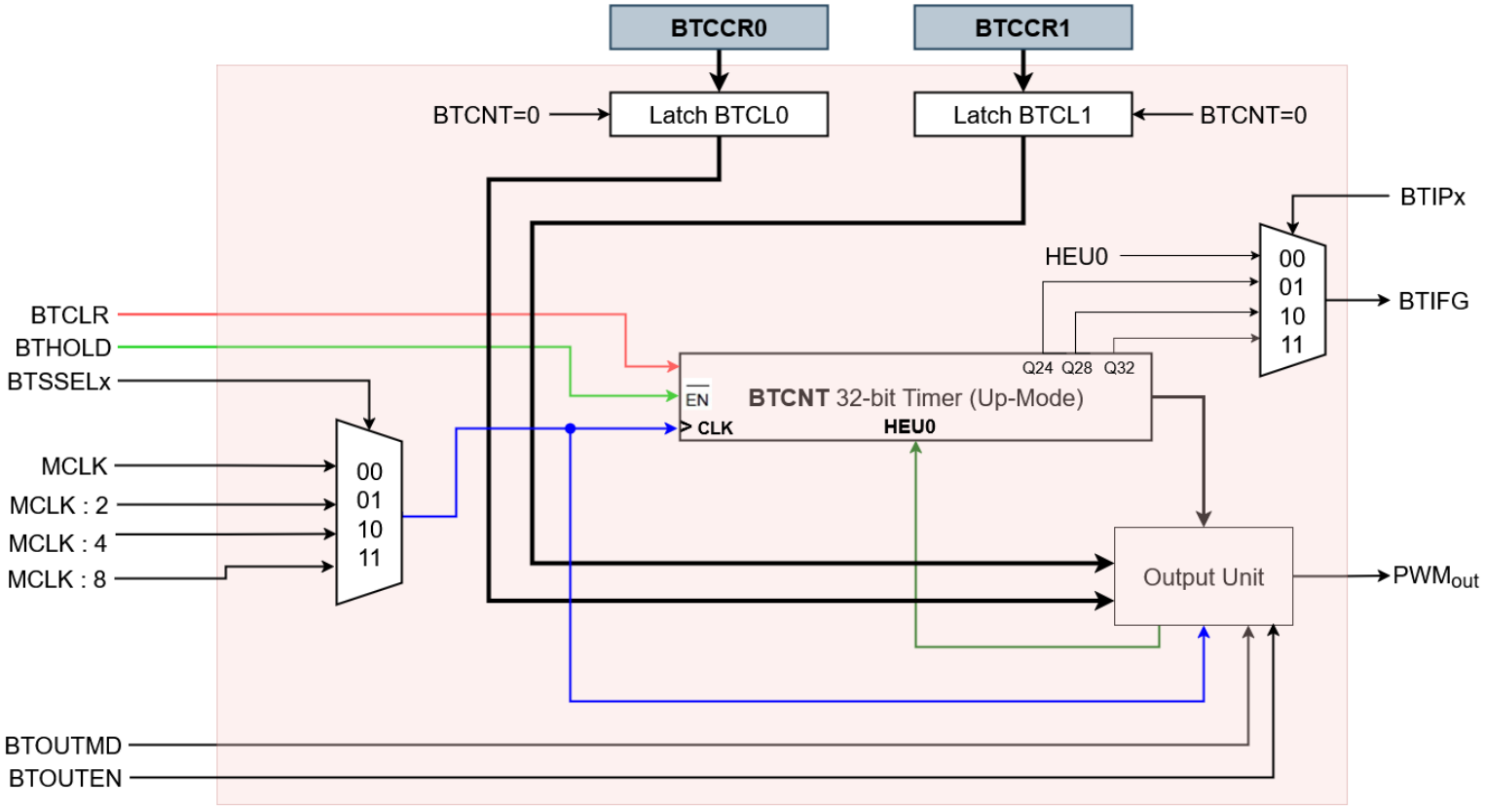
Table 2 : Memory size and latency

4. Required Support of CPU Peripherals

i. **Seven GPIO ports** (six Output and one Input) for peripherals depicted in page 5

ii. **KEY [3-1]:** support array of ***three*** pushbuttons as input device

iii. **Basic Timer with output comparing capabilities:**



BTCTL, Basic Timer Control Register

7	6	5	4	3	2	1	0
BTOUTMD	BTOUTEN	BTHOLD	BTSSELx		BTCLR	BTIPx	
rw	rw	rw	rw	rw	rw	rw	rw

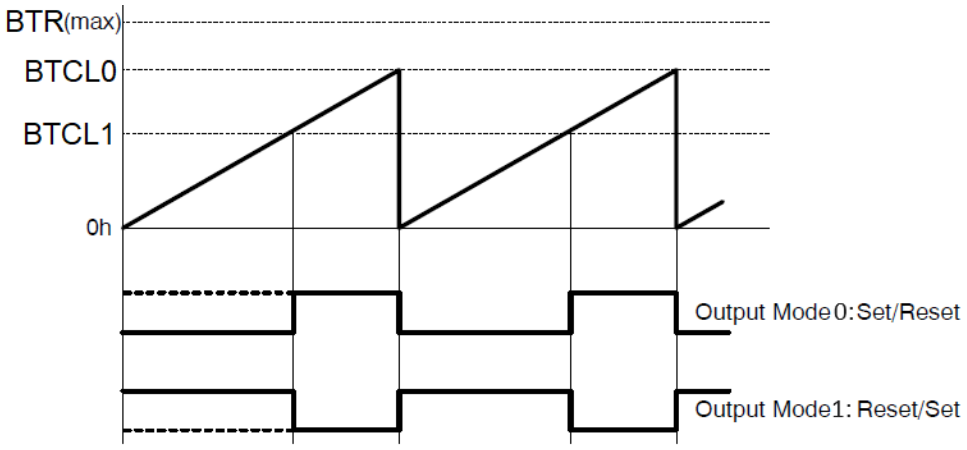
BTCNT, Basic Timer Counter

31	30	. . . . .	1	0
BTCNT				
rw	rw		rw	rw

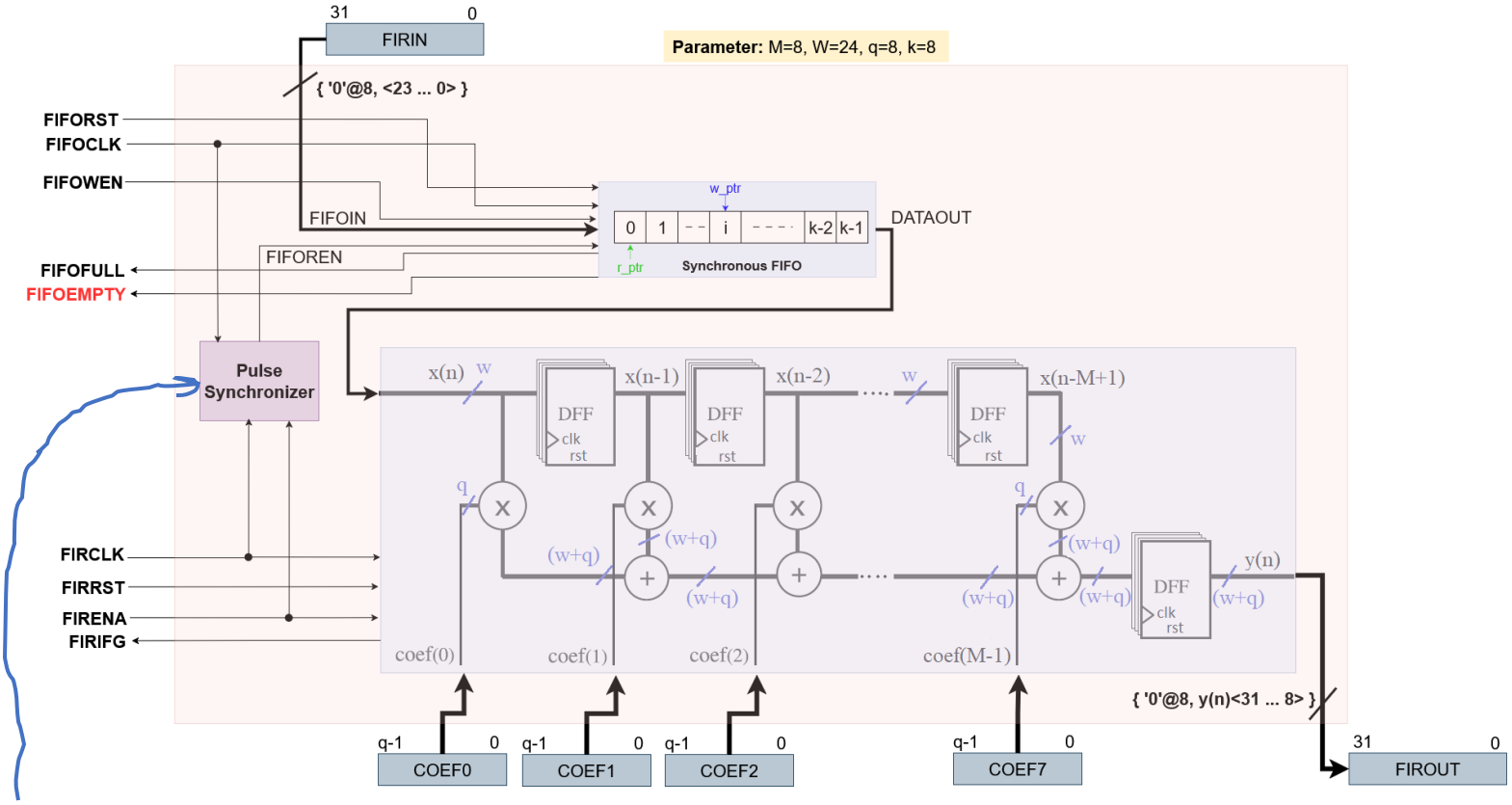
BTCCRx, Basic Timer Compare Register x

31	30	. . . . .	1	0
BTCCRx				
rw	rw	rw	rw	rw	rw	rw	rw

Basic Timer Output compare registers x = {0,1}
Compare data is written to each **BTCCRx** and automatically transferred to **BTCLx**. **BTCLx** holds the data for the comparison to the timer value in the Basic Timer Register, BTCNT.
Note: The register value is zero on RESET.

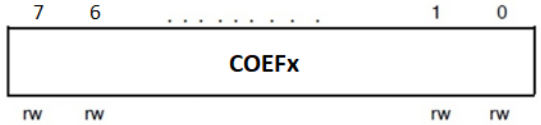


iv. FIR filter HW-Accelerator:

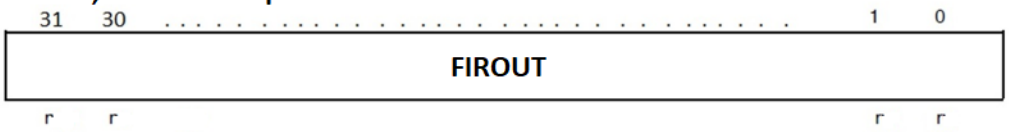


[FIFOREN synchronizer functional diagram]

COEFx , FIR filter coefficient x

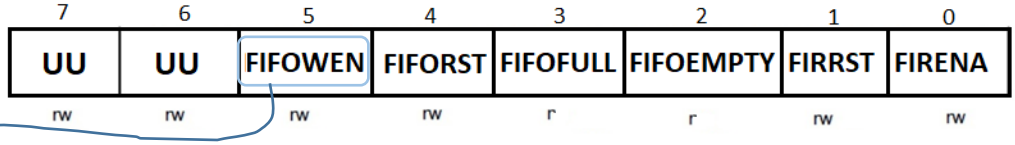


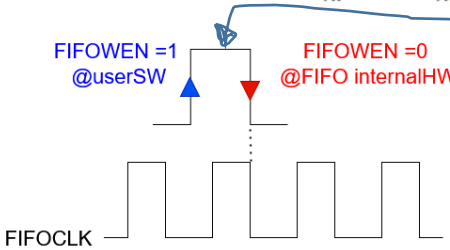
FIROUT, FIR filter output data



FIRIN, FIR filter input data

FIRCTL, FIR filter control





RXBUF, USART Receive Buffer Register

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
r	r	r	r	r	r	r	r

RXBUFx	Bits 7-0	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading RXBUF resets the receive-error bits, and RXIFG.

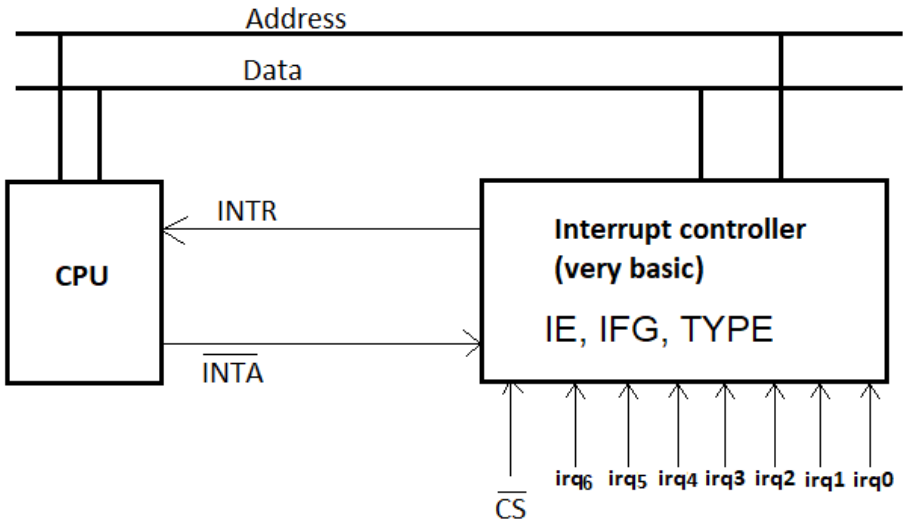
TXBUF, USART Transmit Buffer Register

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw	rw	rw	rw	rw	rw	rw	rw

TXBUFx	Bits 7-0	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted on TXD. Writing to the transmit data buffer clears TXIFG.

Note: for UART module reference, see block diagram of MCUs that use acquainted with.

vi. Interrupt controller:



Notes:

i. The **BTIFG and FIRIFG** flags are reset automatically when the interrupt is serviced.

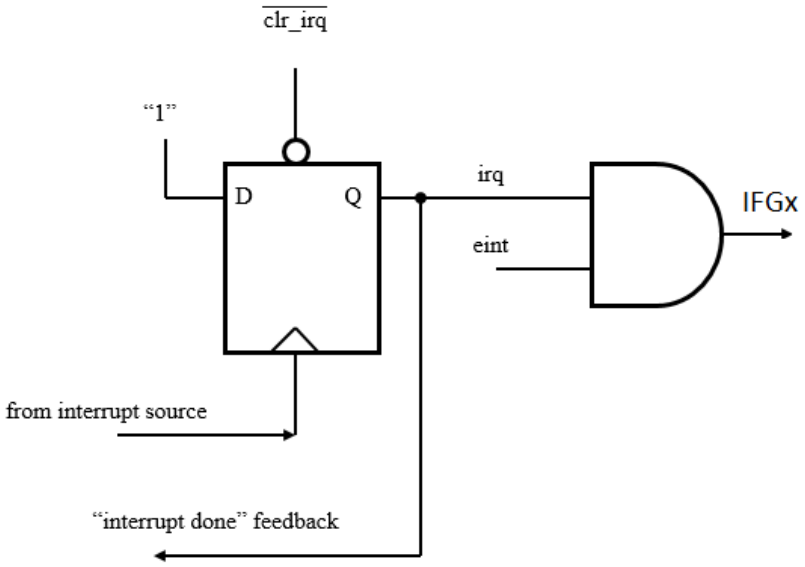
ii. **RXIFG** is automatically reset if the pending interrupt is served or when **RXBUF** is read.

iii. **TXIFG** is automatically reset if the interrupt request is serviced or if a character is written to **TXBUF**

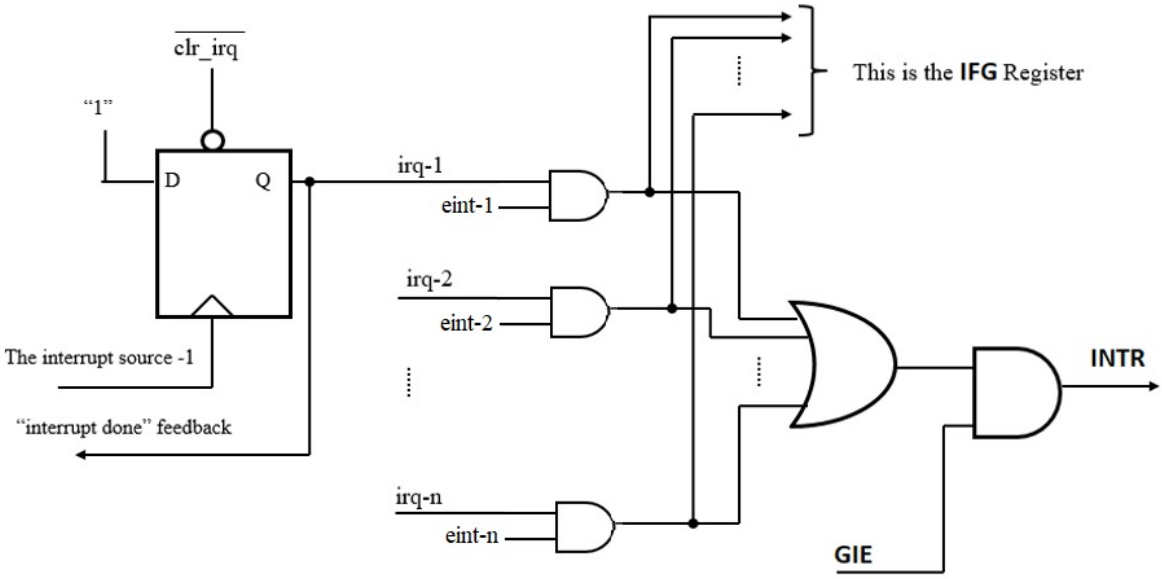
iv. The **KEYiIFG** is reset manually with software (**BTIFG, DIVIFG, RXIFG, TXIFG** as well).

v. As part of CPU services an interrupt, **GIE** is clear (***in HW***) means DINT of other interrupts. Symmetrically, as part of CPU returning from interrupt, **GIE** is set (***in HW***) means EINT of interrupts (back the origin state).

Handling an interrupt:



Handling interrupts from several sources:



IE, Interrupt Enable Register

7	6	5	4	3	2	1	0
0	FIRIE	KEY3IE	KEY2IE	KEY1IE	BTIE	TXIE	RXIE
r-0	rw	rw	rw	rw	rw	rw	rw

IEx Bit x 0 Interrupt not enabled
1 Interrupt enabled

IFG, Interrupt Flag Register

7	6	5	4	3	2	1	0
0	FIRIFG	KEY3IFG	KEY2IFG	KEY1IFG	BTIFG	TXIFG	RXIFG
r-0	rw	rw	rw	rw	rw	rw	rw

IFGx Bit x 0 No interrupt pending
1 Interrupt pending

TYPE, Interrupt Type Register

7	6	5	4	3	2	1	0
0	0	TYPEx					
r-0	r-0	r	r	r	r	r-0	r-0

TYPE Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	RESET	NMI	Highest
04h	UART status error	RXIFG	
08h	UART RX		
0Ch	UART TX	TXIFG	
10h	Basic Timer	BTIFG	
14h	KEY1	KEY1IFG	
18h	KEY2	KEY2IFG	
1Ch	KEY3	KEY3IFG	
20h	FIFOEMPTY	FIRIFG	Lowest
24h	FIROUT		

(Non)-Maskable Interrupt

Maskable Interrupt

*GPIO **without** interrupt capability*

*Peripherals **with** interrupt capability*

```
#            MEMORY Mapped I/O
#-------------------------------------------------------------
#define PORT_LEDR[7-0] 0x800 - LSB byte (Output Mode)
#------------------- PORT_HEX0_HEX1 ---------------------------
#define PORT_HEX0[7-0] 0x804 - LSB byte (Output Mode)
#define PORT_HEX1[7-0] 0x805 - LSB byte (Output Mode)
#------------------- PORT_HEX2_HEX3 ---------------------------
#define PORT_HEX2[7-0] 0x808 - LSB byte (Output Mode)
#define PORT_HEX3[7-0] 0x809 - LSB byte (Output Mode)
#------------------- PORT_HEX4_HEX5 ---------------------------
#define PORT_HEX4[7-0] 0x80C - LSB byte (Output Mode)
#define PORT_HEX5[7-0] 0x80D - LSB byte (Output Mode)
#-------------------------------------------------------------
#define PORT_SW[7-0]   0x810 - LSB byte (Input Mode)
#-------------------------------------------------------------
#define PORT_KEY[3-1]  0x814 - LSB nibble (3 push-buttons - Input Mode)
#-------------------------------------------------------------
#define UTCL            0x818 - Byte
#define RXBF            0x819 - Byte
#define TXBF            0x81A - Byte
#-------------------------------------------------------------
#define BTCTL           0x81C - LSB byte
#define BTCNT           0x820 - Word
#define BTCCR0          0x824 - Word
#define BTCCR1          0x828 - Word
#-------------------------------------------------------------
#define FIRCTL          0x82C - Word
#define FIRIN           0x830 - Word
#define FIROUT          0x834 - Word
#define COEF3_0         0x838 - Word
#define COEF7_4         0x83C - Word
#-------------------------------------------------------------
#define IE              0x840 - LSB byte
#define IFG             0x841 - LSB byte
#define TYPE            0x842 - LSB byte
```

5. Interrupt Service BUS Protocol of a Single-Cycle CPU:

1. **CPU services an interrupt request (**latency of two or three cycles**):**

   This ongoing event is triggered on the falling edge of an INTA signal (the ensuing cycle after INTR is set to '1')

   i. GIE=0 (bit $k0[0] = 0)

   ii. Writing content of register TYPE on Data BUS

   Note: cannot be written on Address BUS because CPU is the only BUS master (executes this protocol).

   iii. Set INTA (INTA='1'), clear BTIFG, DIVIFG flags (in case they were risen)

   iv. Serial emulation execution of ***load*** (of TYPE content) and ***jal*** (to Mem [TYPE] content) where $k1=PC+4

2. **CPU returning from service of an interrupt request (**latency of one cycle**):**

   This event happens as a part of ***reti*** (***jr*** $k1) execution

   GIE=1 (bit $k0[0] = 1) *and* go to return address stored in $k1