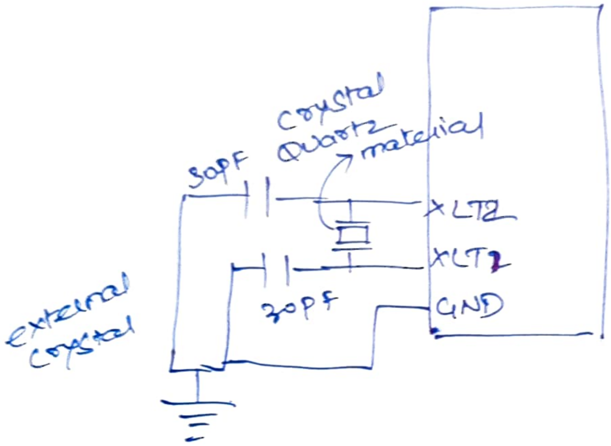
Oscillator & clock generator circuit:-

1 Instruction cycle → complete one instruction/statement.

Ex: Memory cycle
Data fetch cycle
opcode fetch cycle
memory read cycle
memory write cycle & so on...

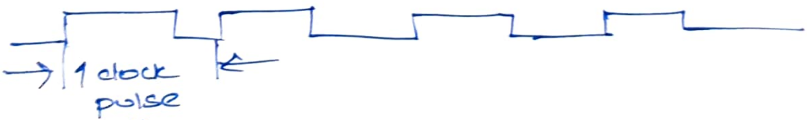


- Inbuilt oscillator
- But source we are connecting (crystal)
- Frequency connected to the crystal, determines speed of the instruction cycle.

Assume, $f_s = 6MHz$ (crystal freq)

clock freq, $f_c = f_s/2 \rightarrow 6/2 = 3MHz$

Oscillator circuit starts generating pulses for the operation.



Time for 1 clock pulse $= 1/f_c = 1/3 = 0.033 \times 10^{-6}$
$= \underline{0.033 \mu s}$

Instruction time is based on the frequency of the crystal connected. When we increase the speed of operation of the MC, we need to increase the freq of the external crystal oscillator.

RST/Reset pin:-

Whateves the operation is done so fas, once the RST botton pressed, then automatically all tke values in the registers are made '0'→zero.

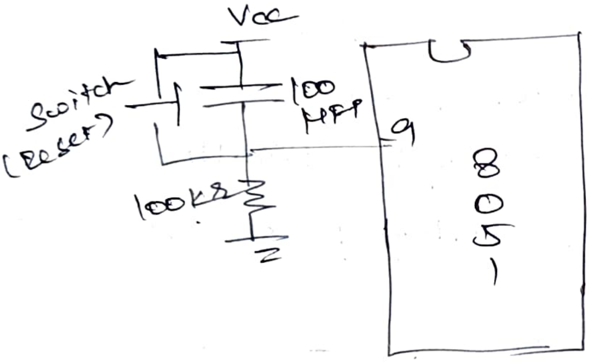
In general, RST = 0 (default) → GND

RST = 1 (if pressed) → +Vcc

For 2 machine cycle, if RST=1, then you can reset all the values to zero.

If we press RST botton in the kit, then it will process the basic operation, some companies they show company details like BIOS.

If RST=1, it makes all tke internal content like address bus, data bus will make to zero, all the registers are zero, stack pointer (07)
↑
initial value



<Power ON RESET CKT>

$\overline{PSEN}$ → program Status Enable.

[4K ROM
128 bytes RAM] internal memory.

If need to used external memory like
64K data memory → ? which to use
64K prog memory → ?
⇓
$\overline{PSEN}$

$\overline{PSEN}$ - 0 ext code memory (ROM)

$\overline{PSEN}$ - 1 ext data memory (RAM)

ALE/PROG

ALE - Address Latch Enable

8bit data → 16 bit address

address Data
(AD0-AD7) ||| (A8 - A15) → Add only (Higher order address)
multiplex

Add/data (lower order address) ←

whenever I am using add, no need of data & vice versa.

ALE → 0 → Data

ALE → 1 → Address

if I am copying code to internal ROM address, in that time if I fire ALE/PROG as +Vcc.

$\overline{EA}$ → External Access

$\overline{EA}$ = 0 (External memory)

$\overline{EA}$ = 1 (Internal memory)

+Vcc & GND → supply pins

+Vcc ↓ 40

GND ↓ 20

4 I/O ports → P0, P1, P2, P3

- 8bits
- used for bidirectional (I/P, O/P)

P3 ⇒ Interrupt, timer } not using Internal means → Alternate functions

P3.0 → RXD → Receiver

P3.1 → TXD → Transmitter

P3.2 → (INT0) Interrup 0 (Ext)

P3.3 → INT1 (Ext) Interrupt 1

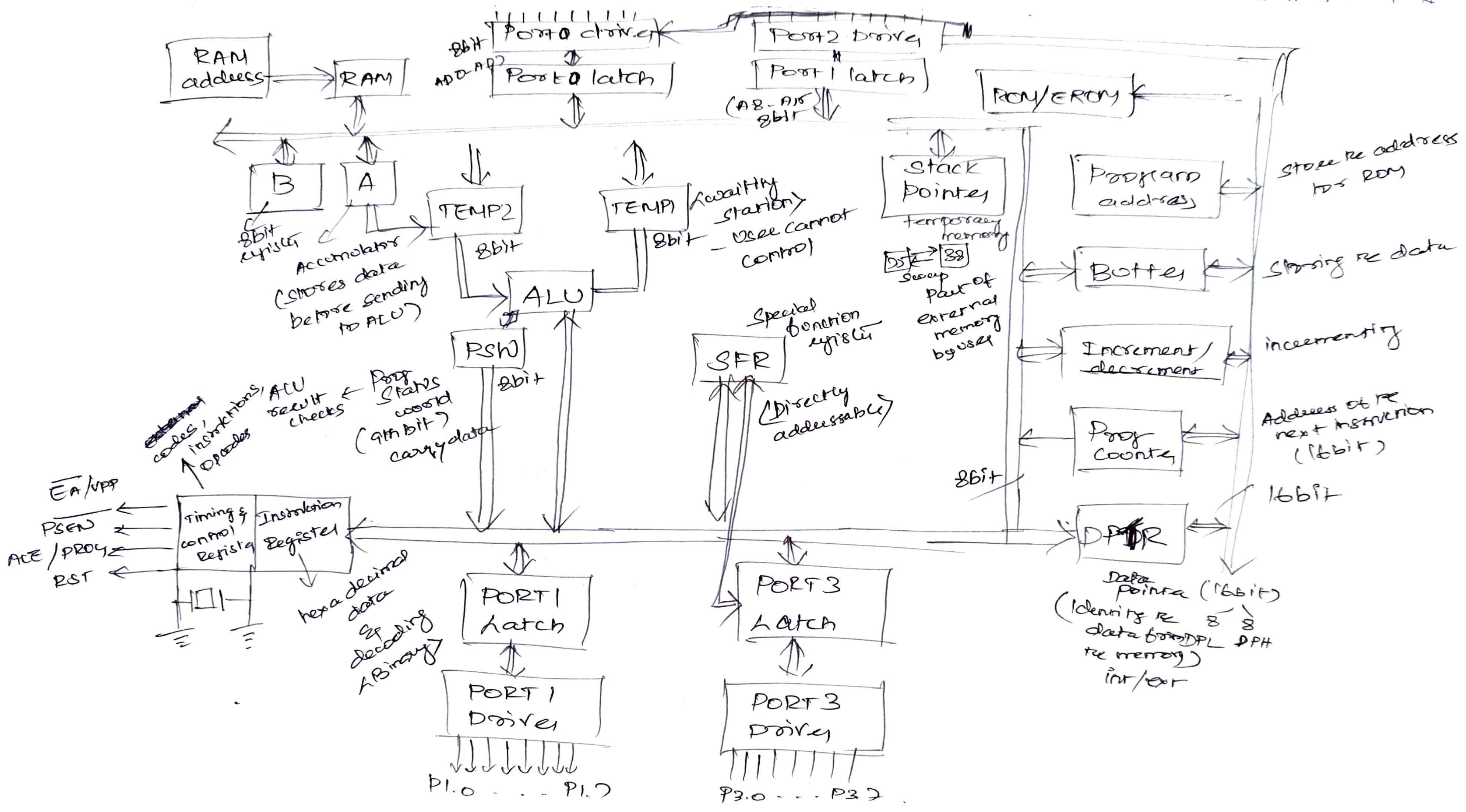
P3.4 → Timer0

P3.5 → Timer1

P3.6 → write line (WR/)

P3.7 → Read line (RD)

Internal architectroee


RAM address
RAM
8bit
AD0-AD7
Port 0 driver
Port 0 latch
Port 2 Driver
Port 1 latch
(A8-A15 8bit
ROM/EROM
B
A
8bit register
Accumulator (stores data before sending to ALU)
TEMP2
8bit
TEMP1
(waiting station)
8bit _ user cannot control
ALU
PSW
8bit
Prog Status word (9th bit) carry data
ALU result checks
~~[illegible]~~ codes, instructions, opcodes
Stack Pointer
temporary memory
DS → 88 Swap
Part of external memory by user
Special function register
SFR
(Directly addressable)
8bit
Program address
Store the address for ROM
Buffer
Storing the data
Increment/ decrement
incrementing
Prog Counter
Address of the next instruction (16bit)
16bit
DP~~T~~R
Data Pointer (16bit)
(Identify the data from the memory) int/ext
8 8
DPL DPH
EA/VPP
PSEN
ALE/PROG
RST
Timing & control Register
Instruction Register
hexa decimal data to decoding to Binary
PORT 1 Latch
PORT 1 Driver
P1.0 . . . P1.7
PORT 3 Latch
PORT 3 Driver
P3.0 . . . P3.7