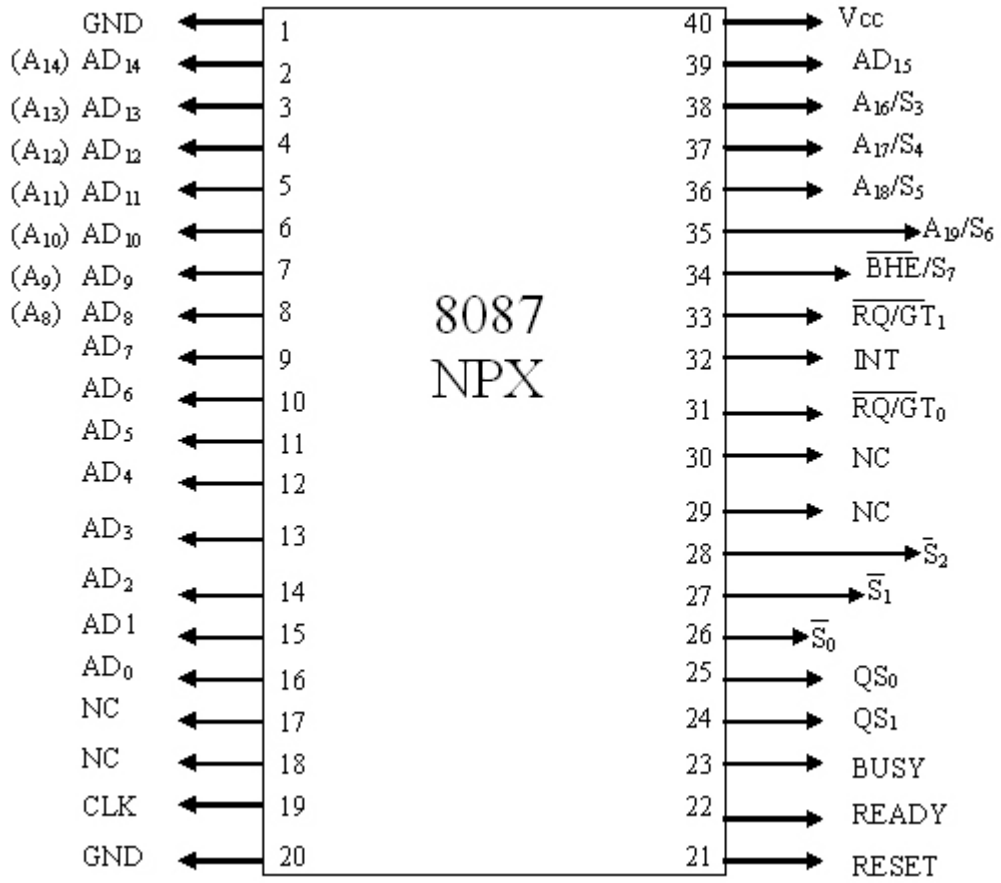
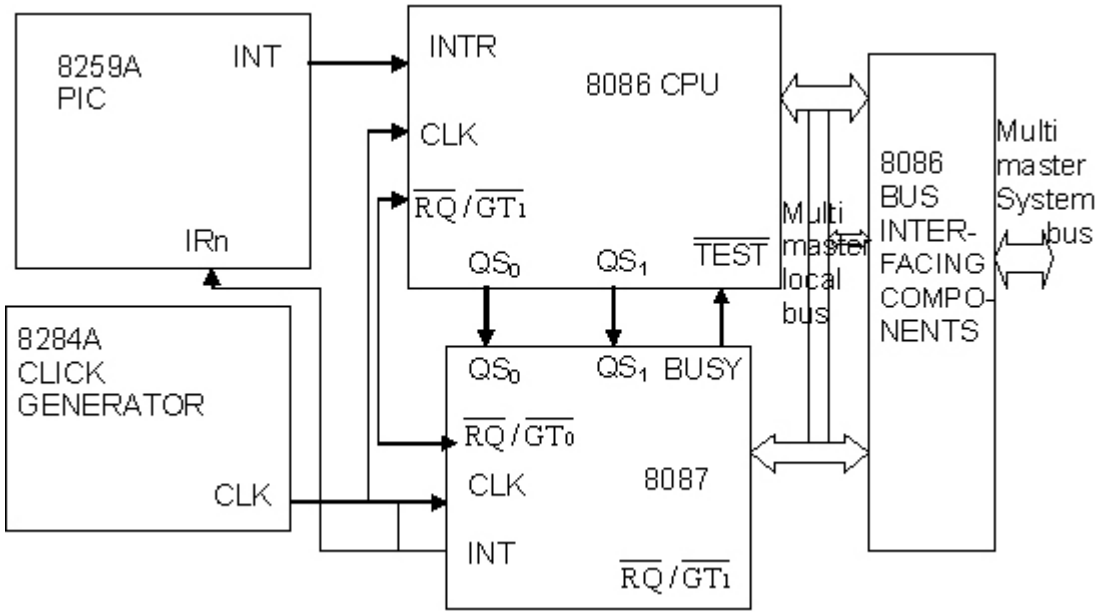
Pin Diagram of 8087



Circuit Connection for 8086 – 8087

- Multiplexed address-data bus lines are connected directly from the 8086 to 8087.
- The status lines and the queue status lines connected directly from 8086 to 8087.
- The Request / Grant signal RQ/GT0 of 8087 is connected to RQ /GT1 of 8086.

- $\overline{\text{BUSY}}$ signal 8087 is connected to TEST pin of 8086.
- Interrupt output INT of the 8087 to NMI input of 8086. This intimates an error condition.



The main purpose of the circuitry between the INT output of 8087 and the NMI input is to make sure that an NMI signal is not present upon reset, to make it possible to mask NMI input and to make it possible for other devices to cause an NMI interrupt.

BHE pin is connected to the system BHE line to enable the upper bank of memory.

The RQ/GT1 input is available so that another coprocessor such as 8089 I/O processor can be connected and function in parallel with the 8087.

One type of Cooperation between the two processors that you need to know about it is how the 8087 transfers data between memory and its internal registers.

When 8086 reads an 8087 instruction that needs data from memory or wants to send data to memory, the 8086 sends out the memory address code in the instruction and sends out the appropriate memory read or memory write signal to transfer a word of data.

In the case of memory read, the addressed word will be kept on the data bus by the memory. The 8087 then simply reads the word of data bus. The 8086 ignores this word .If the 8087 only needs this one word of data, it can then go on and executes its instruction.

Some 8087 instructions need to read in or write out up to 80-bit word. For these cases 8086 outputs the address of the first data word on the address bus and outputs the appropriate control signal.

The 8087 reads the data word on the data bus by memory or writes a data word to memory on the data bus. The 8087 grabs the 20-bit physical address that was output by the 8086.To transfer additional words it needs to/from memory, the 8087 then takes over the buses from 8086.

To take over the bus, the 8087 sends out a low-going pulse on

—— ——

RQ/GT0 pin. The 8086 responds to this by sending another low going pulse back to the

$\overline{RQ}/\overline{GT0}$ pin of 8087 and by floating its buses.

- The 8087 then increments the address it grabbed during the first transfer and outputs the incremented address on the address bus. When the 8087 output a memory read or memory write signal, another data word will be transferred to or from the 8087.
- The 8087 continues the process until it has transferred all the data words required by the instruction to/from memory.
- When the 8087 is using the buses for its data transfer, it sends another low-going pulse out on its RQ/ GT0 pin to 8086 to know it can have the buses back again. The next type of the synchronization between the host processor and the coprocessor is that required to make sure the 8086 hast does not attempt to execute the next instruction before the 8087 has completed an instruction.
- Taking one situation, in the case where the 8086 needs the data produced by the execution of an 8087 instruction to carry out its next instruction.
- In the instruction sequence for example the 8087 must complete the ***FSTSW STATUS*** instruction before the 8086 will have the data it needs to execute the ***MOV AX , STATUS*** instruction.
- Without some mechanism to make the 8086 wait until the 8087 completes the FSTSW instruction, the 8086 will go on and execute the ***MOV AX , STATUS*** with erroneous data .
- We solve this problem by connecting the 8087 BUSY output to the TEST pin of the 8086 and putting on the WAIT instruction in the program.
- While 8087 is executing an instruction it asserts its BUSY pin high. When it is finished with an instruction, the 8087 will drop its BUSY pin low. Since the BUSY pin from 8087 is connected to the TEST pin 8086 the processor can check its pin of 8087 whether it finished it instruction or not.
- You place the 8086 WAIT instruction in your program after the 8087 FSTSW instruction .When 8086 executes the WAIT instruction it enters an internal loop where it repeatedly checks the logic level on the TEST input. The 8086 will stay in this loop until it finds the TEST input asserted low, indicating the 8087 has completed its instruction. The 8086 will then exit the internal loop, fetch and execute the next instruction.

Example

```
FSTSW STATUS ;copy 8087 status word to memory
MOV AX, STATUS ;copy status word to AX to check ; bits
```

(a)

- In this set of instructions we are not using WAIT instruction. Due to this the flow of execution of command will takes place continuously even though the previous instruction had not finished it‟s completion of its work .so we may lost data .

```
FSTSW STATUS ;copy 8087 status word to memory
FWAIT ;wait for 8087 to finish before- ; doing next 8086 instruction
MOV AX,STATUS ;copy status word to AX to check ; bits
```

(b)

- In this code we are adding up of FWAIT instruction so that it will stop the execution of the command until the above instruction is finishes it‟s work .so that you are not loosing data and after that you will allow to continue the execution of instructions.

Another case where you need synchronization of the processor and the coprocessor is the case where a program has several 8087 instructions in sequence.
The 8087 are executed only one instruction at a time so you have to make sure that 8087 has completed one instruction before you allow the 8086 to fetch the next 8087 instruction from memory.
Here again you use the BUSY-TEST connection and the FWAIT instruction to solve the problem. If you are hand coding, you can just put the 8086 WAIT(FWAIT) instruction after each instruction to make sure that instruction is completed before going on to next.
If you are using the assembler which accepts 8087 mnemonics, the assembler will automatically insert the 8-bit code for the WAIT instruction ,10011011 binary (9BH), as the first byte of the code for 8087 instruction.

INTERFACING

Multiplexed address-data bus lines are connected directly from the 8086 to 8087.
The status lines and the queue status lines connected directly from 8086 to 8087.
The Request/Grant signal RQ/GT0 of 8087 is connected to

$\overline{RQ}/GT1$ of 8086.

BUSY signal 8087 is connected to TEST pin of 8086.
Interrupt output INT of the 8087 to NMI input of 8086. This intimates an error condition.

A WAIT instruction is passed to keep looking at its TEST pin, until it finds pin Low to indicates that the 8087 has completed the computation.
SYNCHRONIZATION must be established between the processor and coprocessor in two situations.

a) The execution of an ESC instruction that require the participation of the NUE must not be initiated if the NUE has not completed the execution of the previous instruction.

b) When a processor instruction accesses a memory location that is an operand of a previous coprocessor instruction .In this case CPU must synchronize with NPX to ensure that it has completed its instruction. Processor WAIT instruction is provided.

Exception Handling

The 8087 detects six different types of exception conditions that occur during instruction execution. These will cause an interrupt if unmasked and interrupts are enabled.

1)INVALID OPERATION
2)OVERFLOW
3)ZERO DIVISOR
4)UNDERFLOW
5)DENORMALIZED OPERAND
6)INEXACT RESULT