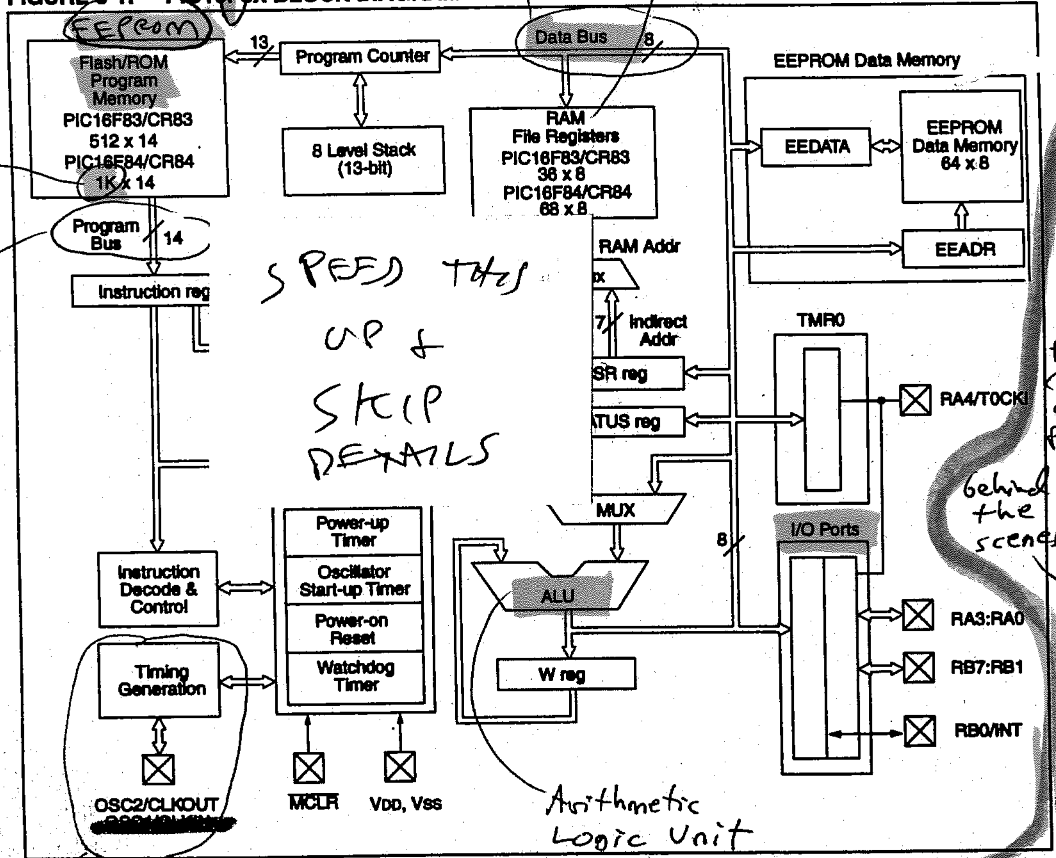
F7.3 (p.265)

"flash" memory

8 bit data bus

Random Access Memory flip-flop (data registers)

FIGURE 3-1: PIC16F8X BLOCK DIAGRAM



1024 MAX ASSEMBLY INSTRUCTIONS

14 bit instructions

Arithmetic Logic Unit

logic (AND, OR) + ARITHMETIC (e.g. full adder circuits)

CAUSES SOMETHING TO HAPPEN ON EVERY CLOCK PULSE

STORES PROGRAM AS HEX CODE (BINARY)

text file CREATED ON PC

BASIC PROGRAM .PBP

PIC BASIC PRO COMPILER

ASSEMBLY LANG. PROGRAM .ASM

ASSEMBLER MPASM

HEX "MACHINE CODE" .HEX

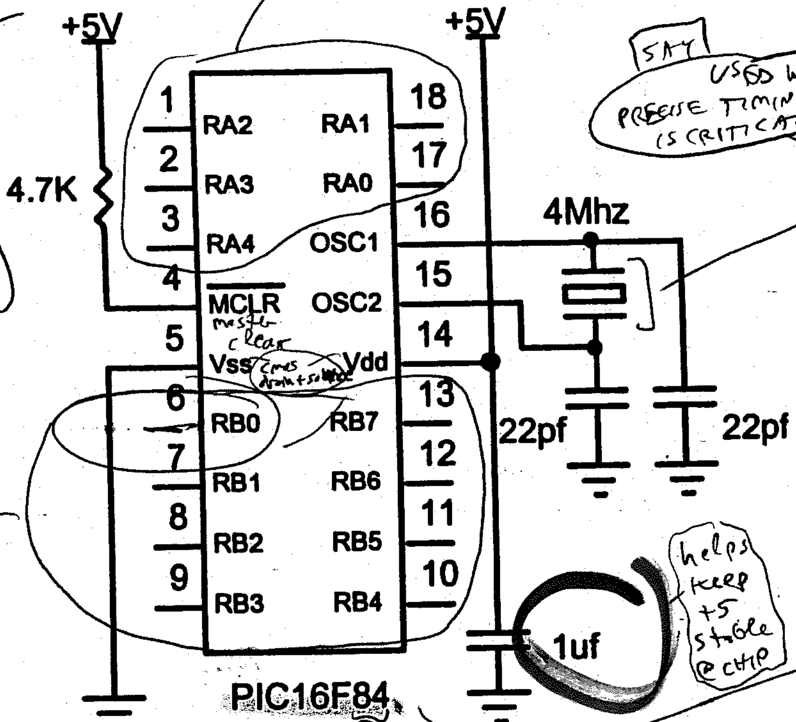
low level chip instructions coded as 0's + 1's

behind the scenes

DEVICES

5 I/O BITS IN PORTA REGISTER

F7.4



ACTIVE LOW

prevents an accidental active-low MCLR (master clear)

master clear

(e.g. from EMI OR STATIC ELECTRICITY if pin left "floating")

SAY USED WHERE PRECISE TIMING IS CRITICAL

QUARTZ CRYSTAL OSCILLATOR (piezoelectric effect + resonance)

ALTERNATIVE INTERNAL TO CHIP w/ MANY PIC (e.g. PIC16F...

helps keep +5 stable @ chip

8 I/O BITS IN PORT B REGISTER (BYTE (8 BITS) OF FF's)

18 PIN DIP

OR 88 OTHERS

MINIMUM REQ'D COMPONENTS

SEE EXAMPLE PIC PROGRAM 1st (IDE C)

: READ IN PicBASIC PRO MANUAL

see on web

LIST ON BOARD

AS EXAMPLE OF WHAT PICBASIC LOOKS LIKE

BE BRIEF

Table 7.5 PicBasic Pro statement summary

Statement	Description
@ assembly statement	Insert one line of assembly language code
ADCIN channel, var	Read the on-chip analog to digital converter (if there is one)
ASM . . . ENDASM	Insert an assembly language code section consisting of one or more statements
BRANCH index, [label1{, label2, . . .}]	Computed goto that jumps to a label based on index
BRANCHL index, [label1{, label2, . . .}]	Branch to a label that can be outside of the current page of code memory (for PICs with more than 2 k of program ROM)
BUTTON pin, down_state, auto_repeat_delay, auto_repeat_rate, countdown_variable, action_state, label	Read the state of a pin and perform debounce (by use of a delay) and autorepeat (if used within a loop)
CALL assembly_label	Call an assembly language subroutine
CLEAR	Zero all variables
CLEARWDT	Clear the watch-dog timer
COUNT pin, period, var	Count the number of pulses occurring on a pin during a period
DATA {@ location,} constant1{, constant2, . . .}	Define initial contents of the on-chip EEPROM (same as the EEPROM statement)
DEBUG item1{, item2, . . .}	Asynchronous serial output to a pin at a fixed baud rate
DEBUGIN {timeout, label,} [item1{,{item2, . . .}]	Asynchronous serial input from a pin at a fixed baud rate
DISABLE	Disable ON INTERRUPT and ON DEBUG processing
DISABLE DEBUG	Disable ON DEBUG processing
DISABLE INTERRUPT	Disable ON INTERRUPT processing
DTMFOUT pin, {on_ms, off_ms,} [tone1{, tone2, . . .}]	Produce touch tones on a pin
{EEPROM {@ location,} constant1{, constant2, . . .}}	Define initial contents of on-chip EEPROM (same as the DATA statement)
ENABLE	Enable ON INTERRUPT and ON DEBUG processing
ENABLE DEBUG	Enable ON DEBUG processing
ENABLE INTERRUPT	Enable ON INTERRUPT processing
~~END~~	Stop execution and enter low power mode
FOR count = start TO end {STEP {-} inc} {body statements} NEXT {count}	Repeatedly execute statements as count goes from start to end in fixed increment
FREQOUT pin, on_ms, freq1{, freq2}	Produce up to two frequencies on a pin
GOSUB label ... Return	Call a PicBasic subroutine at the specified label
GOTO label	Continue execution at the specified label
HIGH pin, LOW	Make pin output high
HSERIN {parity_label,} {time_out, label,} [item1{, item2, . . .}]	Hardware asynchronous serial input (if there is a hardware serial port)
HSEROUT [item1{, item2, . . .}]	Hardware asynchronous serial output (if there is a hardware serial port)
I2CREAD data_pin, clock_pin, control,{ address,} [var1{, var2, . . .}]{, label}	Read bytes from an external I^{2}C serial EEPROM device
I2CWRITE data_pin, clock_pin, control,{ address,} [var1{, var2, . . .}]{, label}	Write bytes to an external I^{2}C serial EEPROM device
IF log_comp THEN label	Conditionally jump to a label
IF log_comp THEN true_statements ELSE false_statements ENDIF	Conditional execution of statements
INPUT pin, OUTPUT	Make pin an input
LCDIN {address,} [var1{, var2, . . .}]	Read RAM on a liquid crystal display (LCD)
LCDOUT item1{, item2, . . .}	Display characters on LCD
{LET} var = value	Assignment statement (assigns a value to a variable)

DO WHILE ... LOOP

REQ'D @ END OF PROGRAM

SKIP

IF LCD MODULE CONNECTED AS SHOWN IN Figure 7.10 IN TEXT BOOK

EDE702 OPTION FOR SERIAL COMM.

ORDER FROM JAMECO OR DIGIKEY

Statement	Description
LOOKDOWN value, [const1{, const2, . . .}], var	Search constant table for a value
LOOKDOWN2 value, {test} [value1{, value2, . . .}], var	Search constant/variable table for a value
LOOKUP index, [const1{, const2, . . .}], var	Fetch constant value from a table
LOOKUP2 index, [value1{, value2, . . .}], var	Fetch constant/variable value from a table
~~LOW pin~~	Make pin output low
NAP period	Power down processor for a selected period of time
ON DEBUG GOTO label	Execute PicBasic debug subroutine at label after every statement if debug is enabled
ON INTERRUPT GOTO label	Execute PicBasic subroutine at label when an interrupt is detected
~~OUTPUT pin~~	Make pin an output
PAUSE period — ms	Delay a given number of milliseconds
PAUSEUS period — μs	Delay a given number of microseconds
{PEEK address, var}	Read byte from a register
{POKE address, var}	Write byte to a register
POT pin, scale, var	Read resistance of a potentiometer, or other variable resistance device, connected to a pin with a series capacitor to ground
PULSIN pin, state, var	Measure the width of a pulse on a pin
PULSOUT pin, period ("one shot")	Generate a pulse on a pin
PWM pin, duty, cycles	Output a pulse width modulated (PWM) pulse train to pin
RANDOM var	Generate a pseudo-random number
RCTIME pin, state, var	Measure pulse width on a pin
READ address, var	Read a byte from on-chip EEPROM
READCODE address, var	Read a word from code memory
RESUME {label}	Continue execution after interrupt handling
~~RETURN~~	Continue execution at the statement following last executed GOSUB
REVERSE pin	Make output pin an input or an input pin an output
SERIN pin, mode,{ timeout, label,} {[qual1, qual2, . . .],}{ item1{, item2, . . .}}	Asynchronous serial input (Basic Stamp 1 style)
SERIN2 data_pin{\flow_pin}, mode, {parity_label,} {timeout, label,} [item1{, item2, . . .}]	Asynchronous serial input (Basic Stamp 2 style)
SEROUT pin, mode, [item1{, item2, . . .}]	Asynchronous serial output (Basic Stamp 1 style)
SEROUT2 data_pin{\flow_pin}, mode, {pace,} {timeout, label,} [item1{, item2, . . .}]	Asynchronous serial output (Basic Stamp 2 style)
SHIFTIN data_pin, clock_pin, mode, [var1{\bits1} {, var2{\bits2}, . . .}]	Synchronous serial input
SHIFTOUT data_pin, clock_pin, mode, [var1{\bits1} {, var2{\bits2}, . . .}]	Synchronous serial output
SLEEP period	Power down the processor for a given number of seconds
SOUND pin, [note1, duration1{, note2, duration2, . . .}]	Generate a tone or white noise on a specified pin
STOP	Stop program execution
SWAP var1, var2	Exchange the values of two variables
TOGGLE pin	Change the state of an output pin
STET ~~WHILE logical_comp statements WEND~~ ≡ DO WHILE Loop	Execute code while condition is true
WRITE address, value	Write a byte to on-chip EEPROM
WRITECODE address, value	Write a word to code memory
XIN data_pin, zero_pin, {timeout, label,} [var1{, var2, . . .}]]	Receive data from an external X-10 type device
XOUT data_pin, zero_pin, [house_code1\key_code1{\repeat1}{, house_code2\key_code2{\repeat2, . . .}]	Send data to an external X-10 type device

STILL USED IN Book

PIC BASIC PRO NOTES

- SEQUENTIAL PROCESSING ⟹

PIC BASIC DOESN'T LOOK AT (EXECUTE) THE NEXT INSTRUCTION UNTIL CURRENT INSTRUCTION IS FINISHED (ex: PAUSE, SOUND, PWM)

EXCEPTION: HARDWARE (H) COMMANDS RUN IN BACKGROUND WHILE PROGRAM CONTINUES (ex: HPWM, HSEROUT)

IF YOU WANT THIS WHILE YOU DO SOMETHING ELSE YOU HAVE TO WRITE YOUR OWN CODE TO DO THIS (IN A LOOP) WHILE DOING OTHER STUFF (OR USE 2ND PIC)

- apostrophe ' : BEGIN COMMENTS OR REMARKS IN YOUR CODE (USE THESE!)

  ex: X=X+1 'increment counter

- colon : : SEPARATOR FOR MULTIPLE STATEMENTS ON 1 LINE

  ex: A=1 : B=2 : C=3

  underscore _ : CONTINUE STATEMENT ON NEXT LINE

  ex: X=Y+Z+_
  A+B

  OR BRANCH LABEL

  ex:
  START:
  ⋮
  IF (X=1) Then start

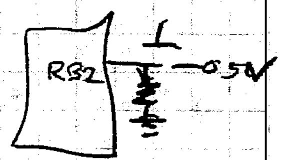
- IMPORTANT REGISTERS

PORTA, PORTB : DIGITAL I/O TO/FROM PIC (1: ~5V, 0: ~0V)

TRISA, TRISB : I/O STATUS OF EACH BIT (PIN) IN PORTA, PORTB (1: INPUT, 0: OUTPUT)

- TO REFER TO AN INDIVIDUAL BIT IN A REGISTER :

REGISTER. BIT



ex: IF (PORTB.2 == 1) THEN ...

- TO SET ALL BITS IN A REGISTER :

REGISTER = %binary [or $hex]

ex: TRISB = %11110000 [or $F0]

⇒ RB0 – RB3 : OUTPUTS
RB4 – RB7 : INPUTS

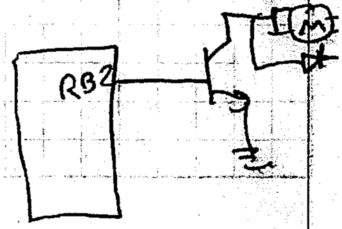
- VARIABLE TYPES (TO STORE THINGS IN YOUR PROGRAM) OR CALC.

Declared w/ : name Var TYPE

name: your name for variable
Var: PICBASIC KEY WORD
TYPE:
- BIT (0 or 1)
- BYTE (8 bits – can rep. #'s from 0 to 255)
- WORD (2 bytes – can rep. #'s fr 0 to 65,535)
- Register. BIT (to name an individual bit in a registe

ex:

motor Var PortB.2
(High motor ' turn on motor



22-141 50 SHEETS
22-142 100 SHEETS
22-144 200 SHEETS
AMPAD