Mnemonics	Operands	Description	Operation
ARITHMETIC AND LOGIC INSTRUCTIONS			
ADDB	Rd, K	Add Constant to Register	Rd <- Rd + K
ADDR	Rd, Rr	Add two Registers	Rd <- Rd + Rr
ANDB	Rd, K	Logical AND Register by Constant	Rd <- Rd & K
ANDR	Rd, Rr	Logical AND two Registers	Rd <- Rd & Rr
DIVB	Rd, K	Divide Register by Constant	Rd <- Rd + K
DIVR	Rd, Rr	Divide two Registers	Rd <- Rd + Rr
DECR	Rd	Decrease Register by one	Rd <- Rd – 1
INCR	Rd	Increase Register by one	Rd <- Rd + 1
MODB	Rd, K	Modulo Register by Constant	Rd <- Rd + K
MODR	Rd, Rr	Modulo two Registers	Rd <- Rd + Rr
MULB	Rd, K	Multiply Register by Constant	Rd <- Rd + K
MULR	Rd, Rr	Multiply two Registers	Rd <- Rd + Rr
NOT	Rd	Logical NOT Register	Rd <- ~Rd
ORB	Rd, K	Logical OR Register by Constant	Rd <- Rd | K
ORR	Rd, Rr	Logical OR two Registers	Rd <- Rd | Rr
RNDB	Rd, K	Random between Register and Constant	Rd <- Rd ? (K - 1)
RNDR	Rd, Rr	Random between Registers	Rd <- Rd ? (Rr - 1)
SUBB	Rd, K	Subtract Constant from Register	Rd <- Rd - K
SUBR	Rd, Rr	Subtract two Registers	Rd <- Rd - Rr
SWAP	Rd, Rr	Swap Rd to Rr and Rr to Rd	Rd <- Rr, Rr <- Rd
XORB	Rd, K	Logical XOR Register by Constant	Rd <- Rd ^ K
XORR	Rd, Rr	Logical XOR two Registers	Rd <- Rd ^ Rr
BANDB	Rd, K	Boolean AND Register by Constant	Rd <- Rd && K
BANDR	Rd, Rr	Boolean AND two Registers	Rd <- Rd && Rr
BNOT	Rd	Boolean NOT Register	Rd <- !Rd
BORB	Rd, K	Boolean OR two Registers	Rd <- Rd || K
BORR	Rd, Rr	Boolean OR Register by Constant	Rd <- Rd || Rr
BIT INSTRUCTIONS			
LSHFTB	Rd, K	Logical Left Shift Register by Constant	Rd <- Rd << K
LSHFTR	Rd, Rr	Logical Left Shift Register by Register	Rd <- Rd << Rr
RSHFTB	Rd, K	Logical Right Shift Register by Constant	Rd <- Rd << K
RSHFTR	Rd, Rr	Logical Right Shift Register by Register	Rd <- Rd << Rr
BRANCH INSTRUCTIONS			
CALL	W	Save program location to stack and Direct Jump	STACK <- PC <- W
JMP	W	Direct Jump	PC <- W
JMPE	W	Direct Jump if Equal	If (F = G) then PC <- W
JMPG	W	Direct Jump if Greater Than	If (F > G) then PC <- W
JMPGE	W	Direct Jump if Greater or Equal	If (F >= G) then PC <- W
JMPL	W	Direct Jump if Less Than	If (F < G) then PC <- W
JMPLE	W	Direct Jump if Less or Equal	If (F <= G) then PC <- W
JMPNE	W	Direct Jump if Not Equal	If (F != G) then PC <- W
JMPMS	W	Direct Jump if specified milliseconds have elapsed.	If (millis>time) PC <- W
JMPUS	W	Direct Jump if specified microseconds have elapsed.	If (micros>time)PC <- W
JMPDE	W	Direct Jump if two data arrays are equal.	If (data=data2) PC <- W
JMPDNE	W	Direct Jump if two data arrays are unequal.	If (data!=data2) PC <-W
RET		Return to location on stack	PC <- STACK
SKPBH	Rr, Rr	Skip next instruction if Register bit is high	If (Rr & (1 << Rr)) skip
SKPBL	Rr, Rr	Skip next instruction if Register bit is low	If (!(Rr) & (1 << Rr)) skip
DATA TRANSFER INSTRUCTIONS			
AREF	K	Analog reference selection of Constant	AREF <- K
DGTLOB	K	Digital write a Constant value	
DGTLOR	Rr	Digital write the Registers value	
DGTLIN	Rd	Digital read into Register	
INB	Rd, K	In Port Constant	Rd <- P
INR	Rd, Rr	In Port Register	Rd <- P
MOVB	Rd, K	Move Constant to Register	Rd <- K
MOVR	Rd, Rr	Move Between Registers	Rd <- Rr
OUT	K, K	Out Port Constant	P <- K

OUTB	Rr, K	Out Port Register	P <- K
OUTBR	K, Rr	Out Port Constant	P <- Rr
OUTR	Rr, Rr	Out Port Register	P <- Rr
POP	Rd	Pop Register from Stack	Rd <- STACK
PUSH	Rr	Push Register on Stack	STACK <- Rr
POPALL		Pop All Registers off Stack	Rs <- STACK
PSHALL		Push All Register on Stack	STACK <- Rs
PNMDB	K	Sets the pin mode of the pin in r0 via Constant	pinMode(r0, K)
PNMDR	Rr	Sets the pin mode of the pin in r0 via Rr	pinMode(r0, Rr)
ANLGIN	Rd	Analog read into register	
ANLGOB	K	Analog write the Register value	
ANLGOR	Rr	Analog write a Constant value	
GET	W	Get byte from Constant address	r1 <- STACK(K + r0)
PUT	W	Put a byte to Constant address	STACK(K + r0) <- r1
DATA	K	Save data array the specified Constant number	Data byte 128
WIPE	W	Null a complete data array at Constant address	Data array = null
MCU CONTROL INSTRUCTIONS			
BEGB	K	Loop Counter Equals Constant. Save Address to LPA	LPAH:LPAL <- PC, LPT<-K
BEGR	Rr	Loop Counter Equals Register. Save Address to LPA	LPAH:LPAL <- PC, LPT<-K
LOOP		Increments LPC by One Compares if Equal to LPT	If (LPC+1 < LPT) P <- LPA
HALTB	K	Pause Execution for Constant Value in Milliseconds	
HALTR	Rr	Pause Execution for Register Value in Milliseconds	
HLTMB	K	Pause Execution for Constant Value in Microseconds	
HLTMR	Rr	Pause Execution for Register Value in Microseconds	
MILLIS	Rr	Put current Millis() to the place pointed.	Rr -> STACK <- Millis
MICROS	Rr	Put current Micros() to the place pointed.	Rr -> STACK <- Micros
NOP		Does nothing	
STOP		Stops program Execution	
SERIAL CONTROL INSTRUCTIONS			
AVAL	Rd	If Serial is Available, Register Returns Bytes in Buffer	If (available) Rd <- BFS
CLEAN		Clears All Waiting Bytes from Serial Buffer	Serial Clear
PRINT	Rr	Prints Register Contents to Serial	Serial Out <- Rr
PRINTCB	K	Prints Constant as Char to Serial	Serial Out Char <- K
PRINTCR	Rr	Prints Register as Char to Serial	Serial Out Char <- Rr
PRINTDC	W	Prints a data array in readable format till null byte	Serial Out Char <- &W
PRINTDB	W	Prints a entire data array in hex.	Serial Out Hex <- &W
READ	Rd	Reads Non-Blocking from Serial into Register	Rd <- Serial Read
SERCTRL	K	Turns Serial On if K > 0 and Off if K = 0	Serial <- K
WREAD	Rd	Reads Blocking from Serial into Register	Rd <- Serial Read

Register Summary		
Address	Description	Specific Usage
0	General Purpose Register	GET/PUT, pin select for analogWrite, digitalWrite, pinMode
1	General Purpose Register	GET and PUT register
2	General Purpose Register	MSB of address for JMPDE/JMPDNE
3	General Purpose Register	Pointer for JMPMS/JMPUS to millis instance returned by MSINIT/USINIT LSB of address for JMPDE/JMPDNE
4	Jump Comparison Register	All Jump Comparisons/ Millis to wait MSB(Most Significant Byte)
5	Jump Comparison Register	All Jump Comparisons / Millis to wait LSB(Least Significant Byte)
6	Stack Pointer	PUSH/POP register
7	Loop Register	Loop until register
8	Loop Register	Loop return address
9	Loop Register	Loop return address
10	Loop Register	Loop counter

Notes:

To select pin mode this is what the numbers represent:

0 = INPUT

1 = OUTPUT

2 = INPUT_PULLUP

To select analog reference this is what the number represent:

0 = DEFAULT

1=INTERNAL

2=EXTERNAL

To compare data spaces/arrays copy one data space/array address into rC and rD,

And the other data space/array address into rE and rF.

To compare, copy the registers in question into registers 4 and 5