로보베이직
명령어 설명서 v2

㈜미니로봇

2006.2

ROBOBASIC

로보베이직

명령어 설명서

v2

등록상표

알림

㈜미니로봇 개발실
http://www.minirobot.co.kr
2006년 2월 작성

차 례

제 1 장 로보베이직 명령어 요약 ····· 7

제 2 장 로보베이직의 일반적인 문법 ····· 14

제 3 장 로보베이직 선언 명령어 설명 ····· 25

DIM ... AS(선언 ... 는) ····· 26

CONST(상수) ····· 28

제 4 장 로보베이직 흐름제어 명령어 설명 ····· 29

IF ... THEN ...(만약 ... 이면 ...) ····· 30

FOR ... NEXT(반복 ... 다음) ····· 33

GOTO(가기) ····· 36

GOSUB ... RETURN(호출 ... 복귀) ····· 37

END(끝) ····· 39

STOP(정지) ····· 41

RUN(실행) ····· 41

WAIT(대기) ····· 42

DELAY(지연) ····· 44

BREAK(중단) ····· 45

제 5 장 로보베이직 디지털 신호 입출력(I/O) 명령어 설명 ····· 46

IN(입력) ····· 47

OUT(출력) ····· 48

BYTEIN(바이트입력) ····· 50

BYTEOUT(바이트출력) ····· 52

INKEY(키입력) ····· 53

STATE(출력상태) ····· 54

PULSE(펄스) ····· 55

TOGGLE(반전) …… 56
KEYIN(키보드) …… 57

제 6 장 로보베이직 메모리관련 명령어 설명 …… 58
PEEK(램읽기) …… 60
POKE(램쓰기) …… 61
ROMPEEK(롬읽기) …… 62
ROMPOKE(롬쓰기) …… 63

제 7 장 로보베이직 LCD모듈 명령어 설명 …… 64
LCDINIT(화면초기) …… 66
CLS(화면지움) …… 67
LOCATE(화면위치) …… 68
PRINT(화면출력) …… 70
FORMAT(형식) …… 72
CSON(커서보임) …… 74
CSOFF(커서감춤) …… 74
CONT(글자밝기) …… 75

제 8 장 로보베이직 모터 제어 명령어 설명 …… 76
ZERO(영점) …… 78
MOTOR(모터설정) …… 81
MOTOROFF(모터해제) …… 83
MOVE(동작) …… 84
SPEED(속도) …… 87
ACCEL(가속) …… 88
DIR(방향) …… 90
PTP(점대점) …… 92
SERVO(서보) …… 95
PWM(펄스폭) …… 96
FASTSERVO(빠른서보) …… 98
HIGHSPEED(고속동작) …… 99

MOVEPOS(동작위치) ····· 100
POS(모터위치) ····· 100
FPWM(주파수펄스) ····· 102
MOVE24(동작24) ····· 103
INIT(초기) ····· 104
MOTORIN(모터입력) ····· 105
AIMOTOR(AI모터설정) ····· 106
AIMOTOROFF(AI모터해제) ····· 108
AIMOTORIN(AI모터입력) ····· 109
GETMOTORSET(모터입력설정) ····· 110
OFFSET(옵셋) ····· 111

제 9 장 로보베이직 음악제어 명령 ····· 112
BEEP(삑) ····· 114
SOUND(소리) ····· 115
PLAY(연주) ····· 117
MUSIC(음악) ····· 120
TEMPO(박자) ····· 123

제 10 장 로보베이직 외부통신 명령어 설명 ····· 124
RX(수신) ····· 127
TX(송신) ····· 129
MINIIN(미니입력) ····· 131
MINIOUT(미니출력) ····· 132
ERX (데이타수신) ····· 133
ETX (데이타송신) ····· 135

제 11 장 로보베이직 아날로그 신호처리 명령어 설명 ····· 137
AD(변환) ····· 138
REMOCON(리모콘) ····· 140
SONAR(초음파) ····· 146
RCIN(조종기입력) ····· 149

GYRODIR(자이로방향) …… 151
GYROSET(자이로설정) …… 153
GYROSENSE(자이로감도) …… 154

제 12 장 로보베이직 처리 명령어와 기타 명령어 설명 …… 155

ON...GOTO(처리...가기) …… 156
RND(난수) …… 158
REMARK(설명) …… 159
ACTION(행동) …… 160
AUTO(자동) …… 161

제 13 장 로보베이직 의사 명령어 설명 …… 164

'$DEVICE('$컨트롤러) …… 165
'$LIMIT('$제한) …… 166

제 1 장

로보베이직

명령어 요약

명령어 요약

로보베이직은 일반적인 베이직 언어 (BASIC Language)에 로봇을 동작하는데 필요한 명령어가 추가된 로봇 제어 전용의 베이직 언어로, ㈜미니로봇에서 개발하여 등록한 소프트웨어이다.

로보베이직의 명령어는, 다음과 같은 명령어로 나뉘어져 있다.

(괄호)안은 한글 명령어이다.
② 표시는 MR-C2000계열 컨트롤러 전용 명령이다.
③ 표시는 MR-C3000계열 컨트롤러 전용 명령이다.

선언/정의에 관련된 명령

DIM(선언)	변수를 선언한다.
AS(는)	변수 선언(DIM) 시에 변수 형을 지정한다.
CONST(상수)	상수를 선언한다.
BYTE(바이트)	변수 선언(DIM) 시에 바이트 형으로 지정한다.
INTEGER(정수)	변수 선언(DIM) 시에 정수 형으로 지정한다.

흐름제어 명령

IF(만약)	조건판단문을 시작한다.
THEN(이면)	조건판단문의 조건이 참이면 다음 실행문을 실행한다.
ELSE(아니면)	조건판단문의 조건이 거짓이면 다음 실행문을 실행한다.
ELSEIF(그렇지않고만약)	다른 조건판단문을 시작한다.
ENDIF(만약끝)	조건판단문을 끝낸다.
FOR(반복)	반복문을 시작한다.
TO(부터)	반복변수에 대한 반복 범위를 지정한다.
NEXT(다음)	반복문을 끝낸다.
GOTO(가기)	무조건 프로그램의 흐름을 분기한다.
GOSUB(호출)	서브루틴(SUB ROUTINE)을 호출한다.

RETURN(복귀) 서브루틴으로부터 복귀한다.
END(끝) 프로그램의 수행을 끝낸다.
STOP(정지) 프로그램의 수행을 일시 정지한다.
RUN(실행) 프로그램의 수행을 계속 실행한다.
WAIT(대기) 명령어의 수행이 끝날 때까지 대기한다.
DELAY(지연) 프로그램의 수행을 일정시간 지연시킨다.
② BREAK(중단) 프로그램의 수행을 일시 중단하고 디버그모드로 전환한다.

디지털 신호 입출력 명령

IN(입력) 입력포트로부터 신호를 읽는다.
OUT(출력) 출력포트로 신호를 내보낸다.
BYTEIN(바이트입력) 바이트 단위의 입력포트로부터 바이트 신호를 읽는다.
BYTEOUT(바이트출력) 바이트 단위의 출력포트로 바이트 신호를 내보낸다.
② INKEY(키입력) 입력포트로부터 키를 입력받는다.
STATE(출력상태) 출력포트의 상태를 알아낸다.
PULSE(펄스) 출력포트로 펄스 신호를 내보낸다.
TOGGLE(반전) 출력포트의 상태를 반전시켜 내보낸다.
③ KEYIN(키보드) 아날로그용 키패드 입력을 받는다.

메모리 관련 명령

PEEK(램읽기) 컨트롤러 내부 RAM(레지스터)의 데이터를 읽는다.
POKE(램쓰기) 컨트롤러 내부 RAM(레지스터)에 데이터를 기록한다.
ROMPEEK(롬읽기) 외부 EEPROM의 데이터를 읽는다.
ROMPOKE(롬쓰기) EEPROM에 데이터를 기록한다.

LCD 관련 명령

LCDINIT(화면초기) LCD모듈을 초기화한다.
CLS(화면지움) LCD모듈의 글자를 모두 지운다.
LOCATE(화면위치) LCD모듈에 글자를 출력할 위치를 설정한다.

	PRINT(화면출력)	LCD모듈에 글자를 출력한다.
	FORMAT(형식)	LCD모듈에 출력할 변수의 형식을 설정한다.
	CSON(커서보임)	LCD모듈에 커서를 나타나게 한다.
	CSOFF(커서감춤)	LCD모듈에 커서를 보이지 않게 한다.
	CONT(글자밝기)	LCD모듈에 나타난 글자의 밝기를 설정한다.
	DEC(십진)	LCD에 변수를 십진수로 출력한다.
	HEX(십육진)	LCD에 변수를 십육진수로 출력한다.
③	BIN(이진)	LCD에 변수를 이진수로 출력한다.

연산 관련 연산자

	AND(그리고)	논리적인 AND 조건식에 사용한다.
	OR(또는)	논리적인 OR 조건식에 사용한다.
	MOD(나머지)	산술 연산 나머지를 구한다.
	XOR(배타합)	논리적인 XOR조건식에 사용한다.
③	NOT(부정)	산술 연산로, 모든 비트를 반전한다.

모터 제어 명령

	ZERO(영점)	서보모터의 영점(기준각도)를 설정한다.
	MOTOR(모터설정)	출력포트에 대해 모터사용여부를 설정한다.
	MOTOROFF(모터해제)	출력포트에 대해 모터사용여부를 해제한다.
	MOVE(동작)	동시에 여러 모터를 동작시킨다.
	SPEED(속도)	모터의 동작속도를 설정한다.
②	ACCEL(가속)	모터 동작의 가속도를 설정한다.
	DIR(방향)	모터 동작방향을 설정한다.
	PTP(점대점)	모터의 동시동작제어 기능 ON/OFF 한다.
	SERVO(서보)	서보모터를 제어한다.
	PWM(펄스폭)	DC모터제어를 위한 펄스폭 제어를 한다.
②	FASTSERVO(빠른서보)	서보모터를 최고의 속도로 동작 시킨다.
③	HIGHSPEED(고속동작)	서보모터의 동작을 빠른 모드로 설정/해제한다.
③	MOVEPOS(동작위치)	POS로 선언된 모터위치 값 그룹을 동작시킨다.

③ POS(모터위치) 로봇의 특정 자세를 설정한다.
③ FPWM(주파수펄스) 펄스의 폭, 주파수를 가변하여 출력한다.
③ MOVE24(동작24) 24개의 서보모터를 동시에 동작한다.
③ INIT(초기) 로봇의 초기 동작 자세를 설정한다.
③ MOTORIN(모터입력) 서보모터의 현재 위치값을 읽어온다.
③ AIMOTOR(AI모터설정) AI모터의 사용을 설정한다.
③ AIMOTOROFF(AI모터해제) AI모터의 사용을 해제한다.
③ AIMOTORIN(AI모터입력) AI모터의 현재 위치값을 읽어온다.
③ SETON(사용설정) 기능 설정의 사용을 설정한다.
③ SETOFF(사용해제) 기능 설정의 사용을 해제한다.
③ ALLON(모두설정) 모든 모터에 대한 기능 설정의 사용을 설정한다.
③ ALLOFF(모두해제) 모든 모터에 대한 기능 설정의 사용을 해제한다.
③ GETMOTORSET(모터입력설정) 현재 서보모터의 값을 읽어들여 현재 상태를 유지한다.
③ OFFSET(옵셋) 현재 설정된 모터의 영점 값을 변경한다.

모터 그룹 지정 파라미터

③ G6A(그룹6A) 서보모터 0번-5번까지 그룹 지정을 위해 사용한다.
③ G6B(그룹6B) 서보모터 6번-11번까지 그룹 지정을 위해 사용한다.
③ G6C(그룹6C) 서보모터 12번-17번까지 그룹 지정을 위해 사용한다.
③ G6D(그룹6D) 서보모터 18번-23번까지 그룹 지정을 위해 사용한다.
③ G6E(그룹6E) 서보모터 24번-29번까지 그룹 지정을 위해 사용한다.
③ G8A(그룹8A) 서보모터 0번-7번까지 그룹 지정을 위해 사용한다.
③ G8B(그룹8B) 서보모터 8번-15번까지 그룹 지정을 위해 사용한다.
③ G8C(그룹8C) 서보모터 16번-23번까지 그룹 지정을 위해 사용한다.
③ G8D(그룹8D) 서보모터 24번-31번까지 그룹 지정을 위해 사용한다.
③ G12(그룹12) 서보모터 0번-11번까지 그룹 지정을 위해 사용한다.
③ G16(그룹16) 서보모터 0번-15번까지 그룹 지정을 위해 사용한다.
③ G24(그룹24) 서보모터 0번-23번까지 그룹 지정을 위해 사용한다.
③ G32(그룹32) 서보모터 0번-31번까지 그룹 지정을 위해 사용한다.

소리 제어 명령

② BEEP(삑) PIEZO에 경고음("삑"소리)을 낸다.
② SOUND(소리) PIEZO에 주파수 음을 발생한다.
② PLAY(연주) PIEZO에 노래를 연주한다.
③ MUSIC(음악) PIEZO에 음악을 연주한다.
③ TEMPO(박자) 음악을 연주할 박자를 설정한다.

외부통신 명령

② RX(수신) RX포트를 이용하여 RS-232신호를 수신한다.
② TX(송신) TX포트를 이용하여 RS-232신호를 송신한다.
② MINIIN(미니입력) 미니통신포트를 이용하여 미니버스신호를 수신한다.
② MINIOUT(미니출력) 미니통신포트를 이용하여 미니버스신호를 송신한다.
③ ERX (데이타수신) RX포트를 이용하여 RS-232신호를 수신한다.
③ ETX (데이타송신) TX포트를 이용하여 RS-232신호를 송신한다.

아날로그 신호처리 명령

③ AD(변환) AD포트에서 아날로그 신호를 읽어온다.
③ REMOCON(리모콘) 적외선 리모콘의 눌린 키 값을 읽어온다.
③ SONAR(초음파) 초음파 포트에서 거리를 읽어온다.
③ RCIN(조종기입력) RC용 조정기의 수신장치로부터 입력값을 읽어온다.
③ GYRODIR(자이로방향) 자이로 센서처리 서보모터의 동작 방향을 설정한다.
③ GYROSET(자이로설정) 서보모터에서 처리할 자이로 센서를 설정한다.
③ GYROSENSE(자이로감도) 자이로 센서의 감도를 설정한다.

처리 명령

ON...GOTO(처리...가기) 변수의 값에 따라 조건분기한다.

기타 명령

	RND(난수)	난수(무작위 수)를 발생한다.
	REMARK(설명)	설명문을 기입한다.
③	ACTION(행동)	정해진 템플릿(templet) 동작을 수행한다.
③	AUTO(자동)	템플릿(templet) 프로그램으로 전환한다.

의사 명령

	'$DEVICE(컨트롤러)	현재 작성하는 프로그램이 동작하는 컨트롤러를 설정한다.
③	'$LIMIT(제한)	모터의 동작 각도 범위를 제한한다.

제 2 장

로보베이직의

일반적인 문법

로보베이직은 일반 베이직 문법에 기초를 두고 있으므로, 많은 부분 베이직 언어와 같거나 유사하다. 여기에서는 로보베이직 일반적인 문법에 대해서 설명한다.

문자집합

로보베이직의 문자 집합은 한글(Micosoft Windows에서 지원하는 모든 한글/한자 포함), 영문자 (A-Z, a-z), 숫자 (0-9) 및 몇 가지의 특수문자로 이루어진다. 또한, 다음에 나타내는 문자들은 로보베이직에서 특별한 의미를 가진다.

문 자	의 미
+	덧셈 기호
-	뺄셈 기호
*	곱셈 기호
/	나눗셈 기호
%	나머지 기호
.	비트 지정기호
&	진수 기호
??	설명문 기호
??	문자열 기호
:	라벨 기호
=	등호(관계연산자) 또는 대입기호
<	부등호(보다 작다)
>	부등호(보다 크다)
<<	비트 좌 쉬프트 기호
>>	비트 우 쉬프트 기호

식과 연산자

식은 상수, 변수, 숫자 등을 연산자를 이용하여 서로 결합해서 구해진 값으로 이루어 질 수 있다. 연산자는 주어진 값에 대해서 산술연산이나 논리연산을 실행한다. 로보베이직에서 연산자는 다음과 같이 분류할 수 있다.

분 류	기 능
산술연산자	산술계산을 한다.
관계연산자	수치를 비교한다.
논리연산자	복합조건을 비교하거나 비트 조작을 실행한다.
비트연산자	비트를 조작하거나 비트에 대한 연산을 행한다.

산술연산자

산술연산자는 산술적인 계산을 하기 위해서 이용되는 기호이며, 덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(% 또는 MOD) 등 일반 베이직 언어에서 사용되는 산술 연산 기호의 사용이 가능하다. 로보베이직에서는 산술연산에 있어 일반 베이직 언어와는 다른 두 가지 주의할 점이 있다.

첫째, ***연산자에 우선순위가 없다.***

산술연산에 연산 순서를 임의로 결정하는 괄호( )를 사용할 수 없으며, 연산자의 순위에 상관없이 사용된 연산자의 순서에 따라 산술 계산한다. 예를 들어 A + B * C 는 일반 베이직의 경우 B와 C를 먼저 곱한 그 결과에 A 를 더하지만, 로보베이직에서는 A와 B를 먼저 더한 다음 그 결과에 C를 곱한다.

예) A = 1, B = 2, C = 3 일 때.

일반 베이직)

A + B * C = 1 + 2 * 3 = 1 + 6 = 7

로보베이직)

A + B * C = 1 + 2 * 3 = 3 * 3 = 9

둘째, ***너무 복잡한 산술계산을 사용하게 되면 예기치 못한 에러가 발생할 수 있다.***

이 경우에는 2-3번 정도 나누어서 계산하면 된다.

산술계산의 예)

D = A * B + C (올바른 사용)
F = A * B / C * D + E (너무 복잡한 계산은 피한다.)

셋째, ***로보베이직은 바이트형, 정수형 만을 지원하기 때문에 나눗셈 연산에서는 결과의 소수점은 무시된다.***

나머지의 연산은 나머지 연산자인 % 또는 MOD를 사용하며, 나머지의 연산자는 몫이 아닌 나머지를 계산결과로서 구한다.

관계연산자

관계연산자는 두 개의 값을 비교할 때 사용된다. 비교의 결과는 "참" (TRUE) 이나 "거짓" (FALSE)이 되며, 이 결과는 IF문(만약) 에서 프로그램의 흐름을 제어하기 위해서 사용된다.

연산자	관 계	식
=	같다.	X = Y
<>	같지 않다.	X <> Y
<	보다 작다.	X < Y
>	보다 크다.	X > Y
<=	보다 작거나 같다.	X <= Y
>=	보다 크거나 같다.	X >= Y

한 개의 식 안에 산술연산자와 논리연산자가 결합되어 있을 경우, 산술연산이 먼저 실행된다.

논리연산자

논리연산자는 복합조건의 비교를 위해 사용하며, 계산결과로서 “참” 이나 “거짓” 을 되돌린다. 이 결과는 IF문(만약)에서 프로그램의 흐름을 제어하기 위해서 사용될 수 있다.

연산자	의 미
AND	논리곱
OR	논리합
XOR	배타적 논리합

각 연산자는 아래에 표시된 것과 같은 계산결과를 갖는다. 표 안에서 "T" 는 참을 나타내고, "F" 는 거짓을 나타낸다.

X, Y 값		논리 연산자가 되돌리는 값 (계산결과)		
X	Y	X AND Y	X OR Y	X XOR Y
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

비트 연산자

비트 연산자는 미니로봇컨트롤러가 입출력포트를 통해 비트 제어를 편리하게 하기 위해 사용하는 변수의 각 비트에 대해서 연산을 행한다. 전체비트에 대한 연산에는 비트합(OR), 비트 곱(AND), 배타적 비트합(XOR)이 있으며, 로보베이직에는 비트를 전체적으로 좌(<<), 우(>>)로 이동(쉬프트) 시키기 위한 연산 기호와 변수의 특정 비트를 지

정하기 위한 "." 연산자가 추가되어 있다.

A의 값이 33 (이진수로 00100001) 이고, B의 값이 15 (이진수로 00001111)이면, 비트 연산자에 의한 다음 표와 같은 결과를 나타낸다.

연산자	결 과
A AND B	1 (00000001)
A OR B	47 (00101111)
A XOR B	46 (00101110)
A << 1	66 (01000010)
A >> 1	16 (00010000)
A.0	1 (A의 0번째 비트)

같은 명령문 안에서 여러 개의 연산자가 사용되고 있을 때 다음과 같은 순서로 연산이 실행된다.

① 산술연산자 / 비트연산자
② 관계연산자
③ 논리연산자

수, 변수/상수 및 기타 문법 설명

로보베이직은 하드웨어 제어를 위해 만들어진 베이직 언어이기 때문에 일반 베이직에서 사용되는 문자열에 관계된 변수나 상수, 변수의 배열형은 지원하지 않는다.

숫자의 형

숫자의 종류에는 바이트형(BYTE), 정수형(INTEGER)이 있으며, 각 숫자의 형에 따른 수의 범위는 표와 같다.

숫 자 형	크 기	범 위
바이트(BYTE)	1 바이트 (8비트)	0-255
정수(INTEGER)	2 바이트 (16비트)	0-65535

로보베이직은 음수를 지원하지 않는다. 그러므로 숫자 앞에 부호 + 또는 -를 첨가하게 되면 잘못된 연산을 수행하거나 에러를 발생하게 된다. 또한 선언된 변수의 경우 연산 도중 숫자의 범위를 벗어나게 되면 에러를 발생하지는 않지만 수행결과가 올바르지 않으므로 항상 적당한 범위의 숫자형으로 선언하여야 한다.

진수

로보베이직은 하드웨어 제어를 위한 언어이기 때문에 일반적으로 사용하는 10진수의 숫자형 표기 외에 16진수 등의 기타 진수로 숫자를 표현하는 것이 프로그램의 작성 시에 편리할 경우가 있다. 로보베이직은 2진수(Bin), 8진수(Oct), 10진수(Dec), 16진수(Hex) 등의 다양한 숫자 표기가 가능하며, 측정 진수표현을 사용하기 위해서는 "&"기호로 진수를 선언하며, 10진수의 숫자형에 대해서는 별도의 선언을 하지 않는다.

10진수로 61의 수를 각 진수에 대해 표현하면 다음 표와 같이 나타낼 수 있고, 진수 표현 시 사용 가능한 숫자의 범위를 벗어나게 되면 에러가 발생하므로, 항상 범위내의 숫자를 사용하여야 한다.

진수	표기	사용가능 숫자	예
2진수	&B	0, 1	&B111101
8진수	&O	0, ... , 7	&O75
10진수	없음	0, ... , 9	61
16진수	&H	0, ... , 9, A, ... F	&H3D

상수와 변수

상수는 프로그램의 수행 중에 값이 변하지 않는 것을 말한다. 로보베이직은 바이트형, 정수형의 숫자에 대한 상수를 정의할 수 있다. 상수의 형은 정의할 때 사용되는 숫자의 범위에 의해 자동 결정되며, 한번 정의된 상수는 다시 정의할 수 없다. 정의된 상수는 사용되는 곳에 정의된 값으로 바로 바뀌기 때문에 프로그램의 크기 등에는 전혀 영향을 미치지 않으며, 자주 사용하는 숫자를 상수로 정의하여 두면 프로그램의 수정이 편리하다.

미니로봇컨트롤러의 하드웨어 제어를 위해 주로 사용되는 상수 정의 예를 들면 다음과 같다.

```
CONST OFF = 0
CONST 모터_1 = 3
CONST 모터_1속도 = 200
```

변수는 프로그램에서 사용할 데이터의 기억장소에 대한 이름을 말한다. 미니로봇컨트롤러에서 사용할 수 있는 변수의 개수는 한정되어 있으므로 사용되는 목적에 따라 변수의 크기를 최소화 할 수 있는 변수형으로 선언하여야 한다.

미니로봇컨트롤러에서 주로 사용되는 변수 선언의 예는 다음과 같다.

```
DIM 모터_1_지연 AS INTEGER
DIM 센서_왼쪽 AS BYTE
```

변수와 상수를 선언할 때는 다음과 같은 규칙을 반드시 지켜야 한다.

첫째, 변수명/상수명에는 한글(한자), 영문, 숫자, "_"등의 문자를 사용할 수 있으며, 첫 문자는 반드시 한글이나 영문이 와야 한다.

둘째, 변수명/상수명의 길이는 최대 64자를 넘을 수 없다.(한글, 영문 구분없이 1글자는 1자로 인식한다.)

셋째, 변수명/상수명은 같은 이름으로 중복 선언할 수 없으며, 대소문자의 구분이 없으므로 주의해서 사용하여야 한다.

넷째, 상수를 선언할 때는 정수형의 범위인 65535 이하의 값을 지정하여야 하며, 이 값을 넘을 경우에는 에러가 발생한다.

비트지정

로보베이직은 사용하는 변수에 대해서 비트 단위로 조작이 가능하다. 비트 단위로 조작하기 위해서는 비트지정 연산자(".")를 사용하며, 바이트형 변수에 대해서는 0-7까지, 정수형 변수에 대해서는 0-15까지의 비트 지정이 가능하다. 비트 지정에는 반드시 숫자나 상수를 사용한다. 비트 지정의 예를 보면 다음과 같다.

```
DIM A AS INTEGER
CONST BIT_2 = 2

A.1 = 1
A.BIT_2 = 0
A.3 = IN(1)          '1번 포트의 값을 읽어 정수형 변수 A의 3번째 비트에 넣음
OUT 2, A.1           '2번 포트에 변수 A의 1번째 비트의 값을 출력
```

설명문

프로그램의 효율적인 작성과 관리를 위해서 프로그램을 작성하는 중간 중간에 프로그램에 대한 설명을 기입해야 할 필요가 있다. 설명문에 대해서는 맨 처음에 ***설명문 기호***

*(') 또는 REMARK(설명) 명령어를 사용*하며, 프로그램에는 수행에는 어떠한 영향도 주지 않는다.

대입문 (=)

대입문은 변수에 값을 대입할 때 사용하며, 단순히 수학에서 사용하는 "="를 사용한다. 대입기호(=)의 좌측에는 항상 변수가 오며, 우측에는 변수, 문자열, 계산식 또는 함수 등이 올 수 있다.

다음은 대입문이 사용되는 몇 가지 예이다.

```
A = B                 '변수끼리 대입
A.1 = 1               '비트지정 대입
A = ADIN(0)           '함수 대입
A = 3 * 2 - 1         '수치 계산식 대입
A = C + B - A         '변수 계산식 대입
A = "1"               '문자의 아스키코드 대입
```

행라벨

행라벨(Label)은 프로그램에서의 위치를 지정할 때 사용하며, 문자나 숫자의 사용이 가능하다. 라벨명을 정할 때는 다음의 몇 가지 규칙이 있다.

첫째, 문자 라벨명은 최대 64자를 넘을 수 없으며, 반드시 첫 자는 한글/영어를 사용하여야 한다.

둘째, 문자 라벨명 뒤에는 반드시 라벨기호(:)를 붙여야 한다.

셋째, 숫자 라벨명은 0~65535 안의 숫자만 가능하며, 라벨기호를 붙이지 않는다.

넷째, 라벨명은 같은 이름으로 중복 선언할 수 없으며, 대소문자의 구분이 없으므로

주의해서 사용하여야 한다.

라벨은 주로 GOTO(가기)나 GOSUB(호출) 등의 프로그램의 흐름제어 명령어에 사용된다. 라벨이 사용된 예를 보면 다음과 같다.

```
DIM A AS INTEGER

START:
        A = IN(0)
        IF A = 0 THEN
                GOTO START
        ELSE
                GOSUB 10
        END
        GOTO START

10      OUT 1, 0
        DELAY 100
        OUT 1, 1
        RETURN
```

제 3 장

로보베이직

선언 명령어 설명

로보베이직에서 변수나 상수를 선언하는데 사용되는 명령어이며, 형 선언자와 같이 사용된다.

DIM ... AS 선언 ... 는

변수를 선언한다.

명령어 구문

① 단일 변수의 선언일 경우 :

- 영문구문 : DIM [변수명] AS [변수형]
- 한글구문 : 선언 [변수명] 는 [변수형]

② 다중 변수의 선언일 경우 :

- 영문구문 : DIM [변수명] AS [변수형], [변수명] AS [변수형], …..
- 한글구문 : 선언 [변수명] 는 [변수형], [변수명] 는 [변수형], …..

명령어 설명

로보베이직에서 사용되는 변수는 먼저 DIM(선언)명령어에 의해서 선언되어야 한다. DIM(선언)명령어에 의한 변수의 선언은 항상 AS(는)를 이용하여 변수형을 선언한다. 선언하고자 하는 변수명에는 영어의 경우 대소문자의 구분이 없으며, 중복하여 같은 변수명으로 선언할 수 없다.

로봇 프로그램에서 센서의 값, 아날로그 변환의 값을 입력받아 값에 따라 처리를 하고자 할 때 변수를 사용한다. 그러므로, 적절한 변수의 사용은 프로그램을 효과적으로 작성할 수 있으며, 로봇 컨트롤러의 종류에 따라 사용할 수 있는 변수의 개수가 정해져 있다.

MR-C2000계열은 총 30바이트 크기의 변수를 사용할 수 있으며, MR-C3000계열은

총 256바이트 크기의 변수를 사용할 수 있다. 바이트형 변수의 경우 1바이트, 정수형 변수의 경우 2바이트 크기를 갖으므로, 변수가 사용되는 범위에 따라 적절히 형 선언을 하여 최대한의 변수를 사용하도록 한다.

명령어 예

```
DIM I AS INTEGER                    '정수형으로 I를 선언
DIM J AS BYTE                       '바이트형으로 J를 선언
선언 K 는 정수                      '정수형으로 K를 선언
선언 모터_1 는 바이트, 모터_2 는 바이트
                                    '바이트형으로 모터_1과 모터_2를 선언
```

CONST 상수

상수를 선언한다.

명령어 구문

- 영문구문 : CONST [상수명] = 숫자
- 한글구문 : 상수 [상수명] = 숫자

명령어 설명

상수로 특정하게 사용되는 숫자에 대해서 간단히 이름으로 구분하면 프로그램의 작성이 편리할 경우에 사용된다. 상수를 사용하면 그냥 숫자 또는 변수를 사용하는 것에 비해서 다음과 같은 이점이 있다.

① 상수는 한번만 정의해주면 프로그램 전체에서 사용이 가능하다.
② 상수는 부주의로 변경되는 경우가 없다.
③ 상수를 사용하면 수정이 쉬워진다.
④ 상수를 사용하는 프로그램은 메모리의 공간을 차지하지 않기 때문에 변수를 사용하는 것보다 효율적인 목적코드를 생성한다.

명령어 예

```
CONST OFF = 0                 'OFF = 0이라는 상수를 선언
CONST A = &HB1001             'A를 십진수 9라는 상수로 선언
상수 참 = 1                   '참 = 1이라는 상수를 선언
상수 거짓 = 0                 '거짓 = 0이라는 상수를 선언
```

제 4 장

로보베이직

흐름제어 명령어 설명

조건을 판단하거나, 반복적인 프로그램의 수행 또는 프로그램 흐름을 제어하기 위해 사용되는 명령어 들이다.

IF ... THEN ... 만약 ... 이면 ...

조건판단을 한다.

명령어 구문

① 단일 조건판단일 경우 :

- 영문구문 : IF *[조건]* THEN
  [조건이 참일 경우의 실행문]
  ENDIF
- 한글구문 : 만약 *[조건]* 이면
  [조건이 참일 경우의 실행문]
  만약끝

② 복합 조건판단일 경우 :

- 영문구문 : IF *[조건1]* THEN
  [조건1이 참일 경우의 실행문]
  ELSEIF *[조건2]* THEN
  [조건2가 참일 경우의 실행문]
  ELSE
  [조건1과 조건2가 아닌 경우의 실행문]
  ENDIF
- 한글구문 : 만약 *[조건1]* 이면
  [조건1이 참일 경우의 실행문]
  그렇지않고만약 *[조건2]* 이면

[조건2가 참일 경우의 실행문]
아니면
[조건1과 조건2가 아닌 경우의 실행문]
만약끝

명령어 설명

IF ... THEN(만약 ... 이면)문이 실행되면 초기 IF(만약)문에서의 조건이 참(TRUE)인지 조사한다. 만약 거짓(FALSE)이면 각각의 ELSEIF(그렇지않고만약) 문의 조건을 검사하거나 ELSE(아니면) 문을 실행한다. 여기서, ELSEIF문은 필요한 만큼 사용할 수 있으며, 사용하지 않은 경우도 많다. 로보베이직의 IF문의 조건이 참인 경우 연결된 THEN(이면)의 다음 명령문장에서부터 다음의 ELSEIF 이나 ELSE 이전의 명령문장까지 실행한 다음, 더 이상의 조건 검사 없이 ENDIF(만약끝)의 다음 명령으로 실행을 옮긴다.

로봇 프로그램에서 없어서는 안될 중요한 구문 중에 하나가 조건(IF..THEN)문이다. 로봇을 외부의 입력에 따라 동작하기 위해서 외부의 입력 값을 변수에 저장한 다음, 변수의 값을 조건문으로 판단하여 값에 따라 동작을 달리하도록 프로그램한다.

명령어 예

① IF 문의 조건식과 실행문이 간단할 경우에는 모두 한 줄에 사용이 가능하다.

```
IF A > 0 THEN B = 5
IF A < 5 THEN B = 0 ELSE B = 1
```

② IF 문의 조건식은 관계연산자에 의해서 두가지 조건을 한꺼번에 사용할 수 있다.

```
IF A > 0 AND A < 5 THEN B = 3
IF A = 7 OR A = 9 THEN B = 1
```

③ 좀 더 복잡한 IF 문의 사용 예를 보면 다음과 같다.

```
IF A = 1 THEN
   B = 2
   C = 3
ELSEIF A = 3 AND A = 5 THEN
   B = 1
   C = 2
ELSEIF A = 8 THEN
   B = 6
   C = 0
ELSE
   B = 0
   C = 0
ENDIF
```

FOR ... NEXT 반복 ... 다음

정해진 수에 따라 반복수행을 한다.

명령어 구문

- 영문구문 : FOR *[반복변수]* = *[시작]* TO *[끝]*
  [반복수행할 실행문]
  NEXT [반복변수]
- 한글구문 : 반복 *[반복변수]* = *[시작]* 부터 *[끝]*
  [반복수행할 실행문]
  다음 *[반복변수]*

명령어 설명

[반복변수]는 반복하는 횟수를 계산하는 변수이며, [시작]은 반복변수의 초기 시작 값이고 [끝]은 반복변수의 마지막 값이다. [시작]과 [끝]에는 숫자, 상수, 변수 등의 사용이 가능하다.

로보베이직은 [반복변수]의 [끝]이 [시작]보다 커야 하며, [반복변수]를 항상 1만큼씩 증가시킨다. 로보베이직에서 FOR...NEXT (반복...다음) 문을 사용할 때 지켜야 할 규칙이 있다.

① FOR...NEXT 문 안에 또 다른 FOR...NEXT 문을 사용할 수 있다.

```
FOR I = 1 TO 10
   FOR J = 1 TO 5
      .....
   NEXT J
```

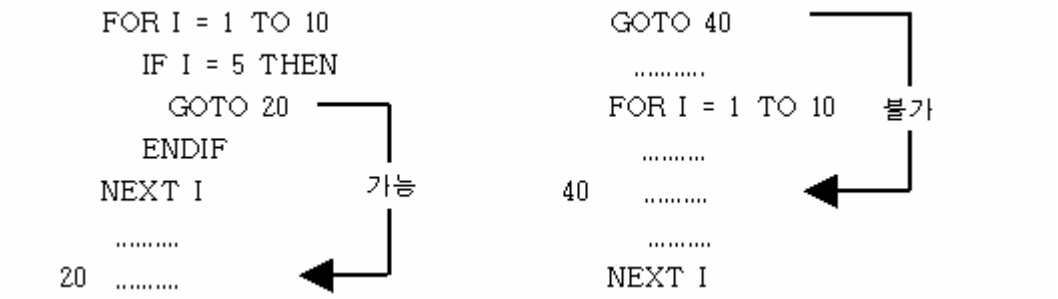
```
NEXT I
```

② 여러 FOR...NEXT 문을 사용할 경우에는 NEXT [반복변수]의 순서를 서로 바꾸어 사용해서는 안 된다.

```
FOR I = 1 TO 10
   FOR J = 1 TO 5
      .....
   NEXT I
NEXT J
```

위의 예제는 NEXT의 반복변수 순서가 바뀌어 있기 때문에 목적코드 생성에는 오류가 발생하지 않지만 미니로봇컨트롤러가 해석하면서 실행해 나갈 때 잘못된 결과가 발생할 수 있다.

③ FOR…NEXT 문의 내부에서 바깥으로 나갈 수는 있지만 바깥에서 FOR…NEXT 문의 내부로 들어올 수는 없다.



④ [반복변수], [시작]과 [끝]에 사용되는 변수의 값을 FOR…NEXT 문 안의 실행문에서 임의로 변경해서는 안 된다.

명령어 예

컨트롤러의 0번 포트에 LED를 연결하고, 5번 깜빡이는 프로그램의 예제이다.

```
DIM A AS BYTE          '반복문에 사용할 변수 선언

FOR A = 1 TO 5         '반복횟수는 1부터 5까지 5번
        OUT 0, 0       '0번 포트의 LED를 ON
        DELAY 100      '일정시간 지연
        OUT 0, 1       '0번 포트의 LED를 OFF
        DELAY 100      '일정시간 지연
NEXT A
```

GOTO 가기

지정한 위치로 무조건 이동한다.

명령어 구문

- 영문구문 : GOTO *[행라벨]*
- 한글구문 : 가기 *[행라벨]*

명령어 설명

GOTO (가기) 명령은 무조건 라벨이 지정된 행으로 분기하며, 행라벨은 프로그램 안에 반드시 선언되어 있어야 한다. GOTO 명령문을 너무 많이 사용하면 프로그램을 이해하기 어려우므로, 가능한 복잡하지 않은 범위 내에서 사용하도록 하며, 되도록 구조화된 흐름제어명령문(IF...THEN문, FOR...NEXT문)을 사용하도록 한다.

명령어 예

```
    DIM I AS INTEGER
    DIM J AS BYTE

    I = 7
    IF I = 6 THEN GOTO L1
      ..............
L1:  J = 1
    OUT I, J
```

GOSUB ... RETURN 호출 ... 복귀

서브루틴을 호출하여 실행한 후 복귀한다.

명령어 구문

- 영문구문 : GOSUB *[행라벨]*

.....

[행라벨]: .....

.....

RETURN

- 한글구문 : 호출 *[행라벨]*

.....

[행라벨]: .....

.....

복귀

명령어 설명

서브루틴(Sub Routine)은 자주 사용되는 프로그램의 부분에 대해서 한번 기술하고 사용될 때마다 호출하여 분기와 복귀를 수행한다. 이러한 서브루틴은 프로그램의 크기를 줄이고, 효율적인 프로그램의 작성을 가능하게 한다.

서브루틴의 내부에서 다른 서브루틴의 호출이 가능하며, *중복 호출이 가능한 횟수는 MR-C2000계열에서 4번이며, MR-C3000계열에서는 5번이다.* 컨트롤러의 한계를 벗어난 중복 호출은 내부의 심각한 에러를 초래하므로 복잡한 중복 호출은 피하도록 한다.

명령어 예

```
        DIM LED_PORT AS INTEGER

        LED_PORT = 1
START: .............
        .............
        GOSUB LED_TOGGLE
        ..............
        GOTO START
        END

LED_TOGGLE:
        TOGGLE LED_PORT
        RETURN
```

END 끝

프로그램의 실행을 끝낸다.

명령어 구문

- 영문구문 : END
- 한글구문 : 끝

명령어 설명

미니로봇컨트롤러는 전원을 인가하면 2초 후에 EEPROM에 저장된 프로그램의 수행을 바로 시작한다. 일반적인 컨트롤러 프로그램의 경우 계속 전원을 인가하는 동안에는 무한히 프로그램의 수행하는 구조로 만들어진다. 그러므로 대부분의 경우 외부의 입력 조건에 따라 조건 수행하는 프로그램으로 구성되어 있지만, 프로그램의 중간에 또는 프로그램의 실행문 다음에 서브루틴 작성 등의 구조적인 프로그래밍을 할 경우에는 프로그램의 수행 끝을 알리는 END(끝)명령을 사용한다.

명령어 예

① 프로그램 수행도중 프로그램의 수행을 끝낸다.

```
        DIM A AS BYTE

START: A = IN(0)
        IF A = 1 THEN END
        ..........
```

```
        GOTO START
```

② 구조적인 서브루틴을 작성할 수 있다.

```
        DIM A AS BYTE

        A = BYTEIN(0)
        IF A = 1 THEN
            GOSUB L1
        ELSEIF A = 3 THEN
            GOSUB L2
        ELSEIF A = 4 THEN
            GOSUB L3
        ELSE
            GOSUB L4
        ENDIF
        ..........
        END

L1: ..............
        RETURN

L2: ..............
        RETURN

L3: ..............
        RETURN

L4: ..........
        RETURN
```

STOP/RUN 정지/실행

프로그램의 수행을 일시정지/계속실행한다.

명령어 구문

- 영문구문 : STOP/RUN
- 한글구문 : 정지/실행

명령어 설명

프로그램의 수행을 일시정지 또는 계속 실행하고자 할 때 사용하며, 일시정지 중일 경우 다시 프로그램을 시작하기 위해서는 RUN(실행) 명령어를 사용하여야 한다.

WAIT 대기

명령어 수행이 끝날 때까지 기다린다.

명령어 구문

- 영문구문 : WAIT
- 한글구문 : 대기

명령어 설명

미니로봇컨트롤러 안에 내장된 컨트롤OS는 타이머를 이용한 실시간처리(Real-Time) 개념의 최신 프로그램이기 때문에 모터의 동작 등 어느 정도의 시간이 소요되는 명령어 실행 시 명령어 처리가 완료될 때까지 프로그램 수행을 정지하지 않고 명령어 실행 시작과 동시에 다음 명령어를 실행한다. 그러나, 명령어의 수행이 완전히 끝난 후 다음 명령어의 실행이 필요로 한 경우 WAIT(대기) 명령어를 사용한다.

로봇의 동작에 사용되는 “MOVE(동작)”명령은 여러 개의 모터가 동시에 구동되기 때문에 모든 모터의 동작이 끝날 때까지 기다리기 위해 많이 사용된다.

명령어 예

다음의 예제는 6개의 모터를 설정각도로 동작시키고 모터가 모두 정지한 후 7,8번이 출력되는 예와 7번은 모터 동작 시 출력되고 8번은 모터정지 후 출력하는 프로그램의 예이다.

예 1)

```
MOVE 120, 100, 140, 90, 70, 150
WAIT
OUT 7, 1
OUT 8, 1
```

예 2)

```
MOVE 120, 100, 140, 90, 70, 150
OUT 7, 1
WAIT
OUT 8, 1
```

DELAY 지연

프로그램 실행을 지정한 시간만큼 지연시킨다.

명령어 구문

- 영문구문 : DELAY *[지연시간]*
- 한글구문 : 지연 *[지연시간]*

명령어 설명

프로그램의 수행을 지정한 시간만큼 지연시키기 위해 사용된다. MR-C2000계열의 컨트롤러는 시간간격이 10ms이며, MR-C3000계열의 컨트롤러는 시간간격이 1ms이다.

[지연시간]에는 숫자, 상수, 변수 등의 사용이 가능하다.

명령어 예

MR-C2000계열의 컨트롤러 사용예)

```
DELAY 10    '100ms동안 지연시킨다. (시간간격 10ms * 10 = 100ms = 0.1초)
```

MR-C3000계열의 컨트롤러 사용예)

```
DELAY 500    '500ms동안 지연시킨다. (시간간격 1ms * 500 = 500ms = 0.5초)
```

BREAK 중단

프로그램의 수행을 일시 중단하고 디버그 모드로 전환한다. 2000

명령어 구문

- 영문구문 : BREAK
- 한글구문 : 중단

명령어 설명

프로그램의 수행을 일시 중단하고 로보베이직의 상태를 디버그모드로 전환하고, 미니로봇컨트롤러의 내부 메모리 상태를 로보베이직 프로그램을 통해 PC로 전송한다. 이때는 반드시 미니로봇컨트롤러와 PC가 인터페이스 케이블로 연결되어 있어야 한다. 만약, 연결되어 있지 않으면 프로그램의 수행을 중단한 상태에서 계속 머물러있게 된다. 디버그 모드에 대한 자세한 설명은 “로보베이직 프로그램 설명”에서 하기로 한다.

“BREAK(중단)” 명령은 MR-C2000계열의 컨트롤러에서만 동작하게 되며, MR-C3000계열에서는 동작하지 않는다. 그러나, MR-C3000계열에서도 디버그 모드를 사용하여 프로그램을 한 단계씩 실행해가면서 프로그램의 진행상황을 추적할 수 있다.

명령어 예

```
.....
BREAK                    '프로그램의 수행을 중단한다.
.....
```

제 5 장

로보베이직

디지털 신호 입출력(I/O)

명령어 설명

미니로봇 컨트롤러는 MR-C2000계열의 경우 최대 12개, MR-C3000계열의 경우 최대 40개의 디지털 신호를 주고 받을 수 있는 입출력 포트(I/O Port)가 있으며, 이 포트를 통하여 다양한 기능을 수행한다. 디지털 신호 입출력 포트에 대해서는 “미니로봇 컨트롤러 설명”을 참조한다.

IN() 입력()

포트로부터 디지털 신호의 값을 읽어들인다.

명령어 구문

- 영문구문 : IN(*[포트번호]*)
- 한글구문 : 입력(*[포트번호]*)

명령어 설명

미니로봇컨트롤러의 포트를 통해서 신호를 입력 받아 값을 변수에 저장한다. 포트에 입력된 값은 0 또는 1의 값이며, 사용되는 변수에는 바이트형, 정수형의 변수 사용이 가능하며 이때는 마지막 0번째 비트의 값만이 유효하다. 가능 효율적인 방법은 바이트형 변수의 비트 지정으로 값을 입력받는다. 포트에 어떤 장치도 연결되어 있지 않는 상태에서 신호의 값을 읽어들이면 전혀 다른 값이 입력될 수 있으므로 주의한다.

명령어 예

```
DIM A AS BYTE
A = IN(0)       '0번 포트에 신호(스위치 또는 센서)를 받아 A 변수에 넣는다.
```

OUT 출력

포트에 디지털 신호를 내보낸다.

명령어 구문

- 영문구문 : OUT *[포트번호], [출력값]*
- 한글구문 : 출력 *[포트번호], [출력값]*

명령어 설명

미니로봇컨트롤러의 포트를 통해서 신호를 내보낸다. 0(LOW)의 값을 내보내면 0V의 신호를 출력하며, 1(HIGH)의 값을 내보내면 +5V의 신호가 출력된다. [출력값]에는 숫자(0 또는 1), 상수, 변수 등의 사용이 가능하며, 출력포트 또한 0 또는 1의 값만이 유효하기 때문에 [출력값]에 변수의 비트 지정을 사용할 수 있다.

명령어 예

다음 예제는 0번포트에 푸시버튼(Push Button)을 연결하고 3번 포트를 LED를 연결하여 포트 입출력을 시험하기 위한 회로에 대해 작성되었다.

```
        DIM A AS INTEGER

        A = 0                       'A 변수를 초기화
START:  A = IN(0)                   '버튼 상태를 읽어들임
                                    'A.0 = IN(0) 로 사용할 수 있다.
        IF A = 1 THEN               '버튼이 안 눌려졌으면
```

```
    OUT 3, 0                    'LED를 끔
ELSE                            '그렇지 않고, 버튼이 눌려졌으면
    OUT 3, 1                    'LED를 켬
ENDIF
GOTO START                      '다시 버튼을 체크
```

BYTEIN() 바이트입력()

바이트 단위의 입력포트로부터 신호를 읽어들인다.

명령어 구문

- 영문구문 : BYTEIN(*[바이트포트번호]*)
- 한글구문 : 바이트입력(*[바이트포트번호]*)

명령어 설명

미니로봇컨트롤러의 포트는 IN(입력)/OUT(출력) 명령어에서처럼 한 비트단위로 신호를 입출력할 수 있지만, 어떤 경우에는 한 단위(여기서는 8개 포트 단위의 바이트포트)로 입출력할 필요가 있다. 미니로봇컨트롤러는 각 포트에 대해서 다음과 같은 바이트포트로 설정되어 있다.

MR-C2000계열의 컨트롤러)

바이트포트	포트
0	0번부터 7번 포트 (0번 포트가 바이트포트의 하위)
1	8번부터 11번 포트 (8번 포트가 바이트포트의 하위)

MR-C3000계열의 컨트롤러)

바이트포트	포트
0	0번부터 7번 포트 (0번 포트가 바이트포트의 하위)
1	8번부터 15번 포트 (8번 포트가 바이트포트의 하위)
2	16번부터 23번 포트 (16번 포트가 바이트포트의 하위)

3	24번부터 31번 포트 (24번 포트가 바이트포트의 하위)
4	32번부터 39번 포트 (32번 포트가 바이트포트의 하위)

BYTEIN(바이트입력)명령어는 미니로봇컨트롤러의 바이트 단위의 입력포트를 통해서 신호를 입력 받아 값을 변수에 저장한다. 변수는 바이트형, 정수형으로 선언되어 있어야 한다.

명령어 예

```
A = BYTEIN(0)          '0번 포트에서 7번 포트의 신호가 모두
                       'A 변수에 입력된다.
```

BYTEOUT 바이트출력

바이트 단위로 포트에 신호를 내보낸다.

명령어 구문

- 영문구문 : BYTEOUT *[바이트포트번호]*, *[출력값]*
- 한글구문 : 바이트출력 *[바이트포트번호]*, *[출력값]*

명령어 설명

미니로봇컨트롤러의 바이트 단위 포트로 값을 내보낸다. *[바이트포트번호]*에는 숫자 또는 상수, 변수 등을 사용할 수 있으며, *[출력값]*에는 0-255사이의 숫자나 상수 또는 바이트 변수를 사용한다.

명령어 예

```
BYTEOUT 0, 255     '0~ 7번 포트에 모두 1이라는 신호(5V)로 내보낸다.
BYTEOUT 0, &h10101010
                   '1,3,5,7번 포트에 신호 1(5V)
                   '0,2,4,6번 포트에 신호 0(0V)를 내보낸다.
```

INKEY 키입력

입력포트로부터 키(스위치) 값을 입력 받는다. 2000

명령어 구문

- 영문구문 : INKEY (*[포트번호]*)
- 한글구문 : 키입력 (*[포트번호]*)

명령어 설명

입력포트를 통해서 버튼과 같은 키를 사용하고자 할 때 사용자는 실제 키를 한번만 누르는 것 같지만 전기적/기계적으로는 수없이 많은 횟수만큼 스위치가 눌려지게 된다. 이러한 현상을 채터링(Chattring)현상이라고 하며, 이는 잘못된 키 값을 읽어올 수 있으므로, 채터링 방지 부가 회로가 필요하다. 미니로봇컨트롤러는 별도의 부가회로 없이 소프트웨어적으로 채터링 방지가 구현되어 있으며, 이를 사용하기 위해서 입력포트에 대해 간단히 키입력(INKEY) 명령어를 사용하면 된다.

명령어 예

0번 포트에 연결된 키(스위치)의 누름 상태를 A변수에 저장한다.

```
DIM A AS BYTE

A = INKEY(0)
```

STATE() 출력상태()

출력포트의 현재 출력상태 값을 읽어온다.

명령어 구문

- 영문구문 : STATE (*[포트번호]*)
- 한글구문 : 출력상태 (*[포트번호]*)

명령어 설명

출력포트를 통해서 신호를 내보낸 후 그 포트의 값의 상태를 알려고 할 때는 다시 입력(IN)함수를 사용해서는 안되고, 출력포트의 상태를 알아내는 출력상태(STATE)함수를 사용하여야 한다.

명령어 예

미니로봇 컨트롤러의 1번 포트의 출력상태를 알아보는 프로그램의 예이다.

```
DIM A AS BYTE

OUT 1, 1
A = STATE(1)          'A = 1
OUT 1, 0
A = STATE(1)          'A = 0
```

PULSE 펄스

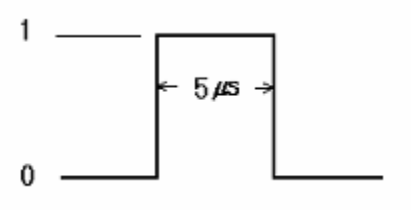
출력포트에 펄스신호를 내보낸다.

명령어 구문

- 영문구문 : PULSE *[포트번호]*
- 한글구문 : 펄스 *[포트번호]*

명령어 설명

출력포트에 아주 짧은 시간(5㎲)동안 펄스 신호를 내보낸다. 펄스신호는 외부에 신호를 주기 위해 사용된다.



*[포트번호]*는 숫자, 상수, 변수 등이 사용 가능하다.

명령어 예

```
PULSE 3                 '3번 포트에 펄스 신호를 내보냄
```

TOGGLE 반전

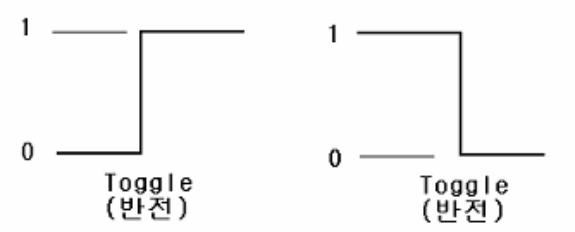
출력포트 신호를 반전한다.

명령어 구문

- 영문구문 : TOGGLE *[포트번호]*
- 한글구문 : 반전 *[포트번호]*

명령어 설명

출력포트의 출력신호를 반전시킨다. 즉, 출력포트 신호가 0(LOW)이면 1(HIGH)로, 1(HIGH)이면 0(LOW)으로 반전된다.



*[포트번호]*는 숫자, 상수, 변수 등이 사용가능하다.

명령어 예

```
OUT 3, 1        '3번 포트에 1신호를 내보낸다.
TOGGLE 3        '3번 포트의 신호를 반전(1에서 0으로)시킨다.
```

KEYIN() 키보드()

여러 개의 키(아날로그 키패드)를 입력받는다. 3000

명령어 구문

- 영문구문 : KEYIN(*[아날로그포트번호], [키갯수]*)
- 한글구문 : 키보드(*[아날로그포트번호], [키갯수]*)

명령어 설명

MR-C3000 계열 컨트롤러의 AD변환 포트(아날로그포트)를 통해 푸시버튼을 최대 16개로 구성된 아날로그 전압 분배회로에 의해 입력된 버튼의 값을 읽어 들인다.

*[아날로그포트번호]*는 0부터 7까지 숫자, 상수 또는 바이트형 변수가 가능하고, *[키갯수]*는 1부터 16까지의 숫자, 상수 또는 바이트형 변수가 가능하다.

읽어들인 버튼의 값은 0부터 16까지이며, 0은 키가 전혀 눌리지 않는 상태이고, 1부터 16은 해당 수는 버튼이 눌린 경우이다.

명령어 예

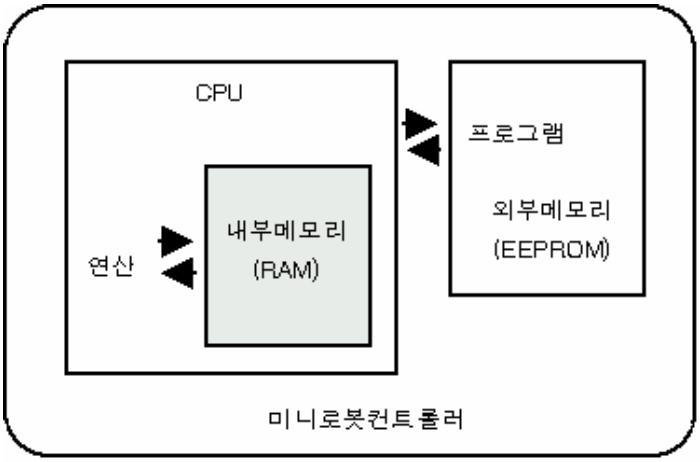
```
DIM K AS BYTE

K = KEYIN(0, 16)    'AD0번포트에 연결된 16개의 키(푸시버튼)로 구성된
                    '아날로그 키패드의 값을 K에 입력한다.
```

제 6 장

로보베이직

메모리관련 명령어 설명

미니로봇컨트롤러는 CPU 및 메모리가 내장되어 있는 초소형 컴퓨터라고 볼 수 있으며, 메모리는 사용자 프로그램의 저장과 연산 등에 필요한 중요한 역할을 수행한다. 미니로봇 컨트롤러의 외부에 부착되어 있는 메모리는 EEPROM으로 주로 사용자의 프로그램을 저장하는데 사용되며, 연산 등을 행하는 메모리는 CPU의 내부에 존재한다.



내부메모리는 RAM이라고 불리며, 미니로봇컨트롤러에서 사용되는 CPU의 특성상 레지스터(Register)라고 불리기도 한다. 내부메모리는 사용자가 프로그램을 작성할 때 사용할 수 있는 변수의 개수 등에 관련이 있으며, *MR-C2000계열 컨트롤러의 경우 30바이트의 변수공간*을 갖고 있다. 이는 바이트 형으로 선언된 변수의 경우 30개를, 정수형(2바이트 차지)으로 선언된 변수만 사용하게 되면 15개까지 사용가능 함을 나타낸다. *MR-C3000계열의 경우 256바이트의 변수공간*을 가지고 있다. 내부메모리의 다른 영역은 미니로봇컨트롤러가 내부적으로 사용함으로 함부로 데이터를 건드려서는 안 된다.

외부메모리(EEPROM)는 사용자가 프로그램을 작성할 때 작성되는 프로그램의 크기에 관련이 있으며, MR-C2000 계열의 경우 4K바이트의 메모리를 가지고 있으며, MR-C3000 계열의 경우 모델에 따라 12K바이트, 32K바이트, 64K바이트의 메모리를 가지고 있다.

PEEK() 램읽기()

내부 메모리의 내용을 읽어온다.

명령어 구문

- 영문구문 : PEEK (*[램영역]*)
- 한글구문 : 램읽기 (*[램영역]*)

명령어 설명

PEEK(램읽기) 함수는 내부 메모리 내의 데이터를 가져오지만, 정확한 내부메모리의 구조를 알지 못하면, 사용하지 못하므로 함부로 건드리지 않는다.
*[램영역]*은 컨트롤러에 따라 사용하는 영역이 다르다.

MR-C2000계열의 컨트롤러)
0-255 사이의 숫자, 상수, 바이트형, 변수를 사용할 수 있다.

MR-C3000계열의 컨트롤러)
0-255 사이의 숫자, 상수, 바이트형, 변수를 사용할 수 있다.

명령어 예

```
DIM A AS BYTE

A = PEEK(43)          '램영역 43번지의 값을 A 변수에 불러온다.
```

POKE 램쓰기

내부메모리에 데이터를 기록한다.

명령어 구문

- 영문구문 : POKE *[램영역]*, *[데이터]*
- 한글구문 : 램쓰기 *[램영역]*, *[데이터]*

명령어 설명

미니로봇컨트롤러의 내부 메모리에 데이터를 기록하고자 할 때 POKE(램쓰기) 명령어를 사용한다. PEEK(램읽기) 함수는 데이터를 읽기만 하지만, POKE(램쓰기) 명령어는 데이터를 직접 써넣을 수 있기 때문에 상당한 신중을 기해야 한다.

*[램영역]*은 MR-C2000계열 컨트롤러의 경우 0-255 사이의 숫자, 상수, 바이트형 변수를 사용할 수 있으며, MR-C3000계열 컨트롤러의 경우 0-65535 사이의 숫자, 상수, 바이트형 변수를 사용할 수 있다. *[데이터]*는 숫자, 상수, 변수(정수형 변수 포함)를 사용할 수 있다.

명령어 예

POKE &h40, 100 *'램영역 40번지(16진수)에 100의 값을 기록한다.*

ROMPEEK() 롬읽기()

외부 EEPROM의 데이터를 읽어온다.

명령어 구문

- 영문구문 : ROMPEEK (*[롬영역]*)
- 한글구문 : 롬읽기 (*[롬영역]*)

명령어 설명

미니로봇컨트롤러는 사용자 프로그램의 저장이나 특수한 목적으로 사용되는 EEPROM을 사용한다. 외부메모리를 제어하는 ROMPEEK(롬읽기)함수나 ROMPOKE(롬쓰기)명령어는 사용자가 영구적으로 저장하고 싶은 데이터를 저장하는데 사용할 수 있다. 그러나, 이미 예약되어 있는 영역을 사용하게 되면 치명적인 오류가 발생할 수 있다. *[롬영역]*은 숫자, 상수, 변수를 사용할 수 있다.

ROMPOKE 롬쓰기

외부 EEPROM에 데이터를 기록한다.

명령어 구문

- 영문구문 : ROMPOKE *[롬영역], [데이터]*
- 한글구문 : 롬쓰기 *[롬영역], [데이터]*

명령어 설명

미니로봇컨트롤러는 사용자 프로그램의 저장이나 특수한 목적으로 사용되는 EEPROM을 사용한다. 외부메모리를 제어하는 ROMPEEK(롬읽기)함수나 ROMPOKE(롬쓰기)명령어는 사용자가 영구적으로 저장하고 싶은 데이터를 저장하는데 사용할 수 있다. 그러나, 이미 예약되어 있는 영역을 사용하게 되면 치명적인 오류가 발생할 수 있다. *[롬영역]*은 숫자, 상수, 변수를 사용할 수 있다. *[데이터]*는 0-255사이의 숫자나 바이트형 변수, 상수를 사용할 수 있다.

제 7 장

로보베이직

LCD모듈 명령어 설명

미니로봇 컨트롤러에서 사용하는 LCD 모듈은 아래 그림의 MR-16202와 같은 정해진 형태에 대해서만 사용이 가능하다.

MR-C2000계열의 컨트롤러는 6번 포트에 LCD모듈을 연결하고, MR-C3000계열의 컨트롤러는 정해진 LCD 포트에 LCD 모듈을 연결하여야 한다.

여기에서 설명하는 LCD 관련 명령어들은 사용자가 원하는 문자열을 표시하거나, LCD모듈을 제어할 수 있는 명령어 들이다.

MR-16202 LCD모듈

LCDINIT 화면초기

LCD모듈을 초기화한다.

명령어 구문

- 영문구문 : LCDINIT
- 한글구문 : 화면초기

명령어 설명

LCD모듈을 사용하기 전에 항상 LCD모듈을 초기화하여야 한다. 초기화하지 않은 LCD모듈을 이용하여 문자열을 출력하면 원하지 않는 이상한 문자가 나타날 수 있으므로, 사용하기 전 프로그램의 처음 부분에 항상 LCDINIT(화면초기) 명령어를 사용한다.
LCD모듈을 초기화하면 LCD화면의 모든 글자를 지우고, 커서를 맨 첫 위치로 보낸다.

명령어 예

```
LCDINIT          'LCD모듈을 초기화한다.
```

CLS 화면지움

LCD모듈의 화면을 지운다.

명령어 구문

- 영문구문 : CLS
- 한글구문 : 화면지움

명령어 설명

LCD모듈의 나타나있는 글자들을 모두 지우고자 할 때 CLS(화면지움) 명령어를 사용한다. CLS(화면지움) 명령어는 모든 글자를 지우고, 커서의 위치를 처음으로 되돌린다. LCDINIT(화면초기) 명령어와의 차이점은 CLS(화면지움) 명령어는 단지 LCD모듈의 표시되는 글자만을 지우지만, LCDINIT(화면초기) 명령어는 LCD모듈을 초기화함으로 LCD모듈 내부의 C.G.RAM, 글자밝기 값을 기본값으로 설정, 그리고 LCD모듈 상에 나타나지 않는 내부 변수 들을 모두 초기화한다.

명령어 예

```
CLS          'LCD모듈 화면의 내용을 지운다.
```

LOCATE 화면위치

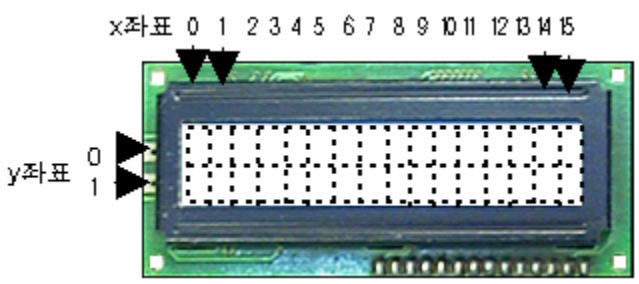
LCD모듈에 출력할 글자의 위치를 지정한다.

명령어 구문

- 영문구문 : LOCATE *[x좌표], [y좌표]*
- 한글구문 : 화면위치 *[x좌표], [y좌표]*

명령어 설명

LOCATE(화면위치)명령어는 LCD모듈의 원하는 위치에 문자를 출력하고자 할 때 x좌표와 y좌표에 대해 위치를 설정한다.



16x2라인의 LCD모듈에 대한 좌표는 위 그림과 같이 결정되어 있으며, 다른 LCD모듈에 대해서도 마찬가지 방법으로 좌표가 결정되어 있다. x, y좌표는 숫자 또는 상수, 변수의 사용이 가능하며, *항상 0부터 시작*함에 주의한다.

명령어 예

```
LOCATE 0, 0        'LCD모듈의 커서를 처음위치로 옮긴다.
LOCATE 4, 1        'LCD모듈의 커서를 (4, 1)과표로 옮긴다.
```

PRINT 화면출력

LCD모듈에 글자를 출력한다.

명령어 구문

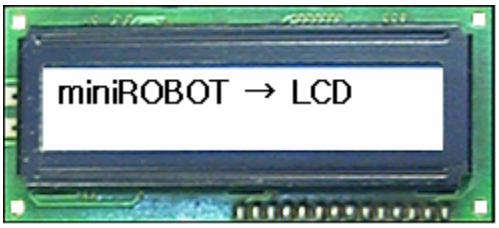
- 영문구문 : PRINT *"[문자열]", [숫자]/ "[문자열]"*, ...
- 한글구문 : 화면출력 *"[문자열]", [숫자]/ "[문자열]"*, ...

명령어 설명

PRINT(화면출력)명령어는 LCD모듈의 현재 커서위치에 문자를 출력한다. *[문자열]*은 큰 따옴표 (")로 구분한다. *[숫자]*는 1~255사이(0은 사용할 수 없다.) 의 숫자나 상수를 사용할 수 있고, LCD모듈에는 숫자의 아스키 코드(ASCII)에 해당하는 문자가 출력된다. 한글 LCD모듈을 사용할 때는 *[문자열]*에 한글을 사용할 수 있다.

명령어 예

```
CLS
PRINT "miniROBOT ", 126, " LCD"
```



16x2라인 영문 LCD모듈에 아스키 코드로 문자를 출력할 경우에 나타나는 문자들의 집합을 나타내면 다음과 같다.

각 코드에 해당되는 문자들은 LCD 모듈의 종류에 따라 달라질 수 있다.

FORMAT() 형식()
DEC() 십진()
HEX() 십육진()
BIN() 이진()

LCD모듈에 표시할 숫자의 형식을 지정한다.

명령어 구문

- 영문구문 : FORMAT (*[변수], [출력형식], [자릿수]*)
- 한글구문 : 형식 (*[변수], [출력형식], [자릿수]*)

명령어 설명

LCD모듈에 변수를 출력하고자 할 때는 일반 문자열을 출력하는 것과는 달리, 일정한 형식에 맞추어서 출력하여야 한다. FORMAT(형식) 명령어는 반드시 PRINT(화면출력) 명령어 뒤에 사용되며, 변수에 대한 출력형식을 지정한다.

[변수형]/[출력형식]	기본 자릿수	표현 수
바이트형/십진	3	0 ~ 255
바이트형/십육진	2	00 ~ FF
바이트형/이진	8	00000000 ~ 11111111
정수형/십진	5	0 ~ 65535
정수형/십육진	4	0000 ~ FFFF

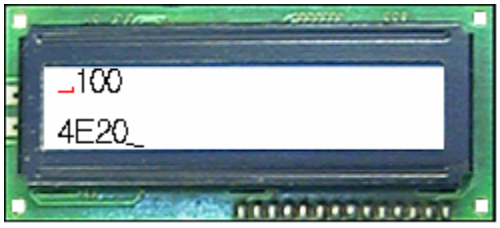
정수형/이진	16	0000000000000000 ~ 1111111111111111

*[변수]*에는 바이트형 또는 정수형 변수를 사용할 수 있으며, *[출력형식]*에는 출력하려는 변수 값을 10진수나 16진수로 출력하기 위한 DEC(십진) 또는 HEX(십육진), BIN(이진) 예약어를 사용한다. *[자릿수]*에는 출력변수에 대한 자릿수를 지정하며, 생략 가능하다. *[자릿수]*를 생략하면, *[변수]*와 *[출력형식]*에 따라 기본값이 다르다.

*[자릿수]*가 출력숫자의 개수보다 클 경우에는 오른쪽맞춤에 의해 출력되고, 빈 칸은 공란(Space)을 출력한다. *[자릿수]*가 출력숫자의 개수보다 작을 경우에는 맨 앞자부터 *[자릿수]*의 수만큼 출력된다.

명령어 예

```
DIM A AS BYTE
DIM B AS INTEGER

LCDINIT
A = 100
B = 20000
LOCATE 0, 0
PRINT FORMAT(A, DEC, 4)
LOCATE 0, 1
PRINT FORMAT(B, HEX)
```



CSON() 커서보임

CSOFF() 커서감춤

LCD모듈에 커서를 나타내거나/보이지 않게 한다.

명령어 구문

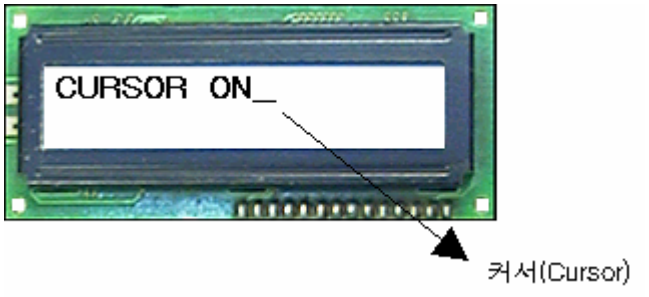
- 영문구문 : CSON / CSOFF
- 한글구문 : 커서보임 / 커서감춤

명령어 설명

LCD모듈에 문자열을 출력할 때 커서를 보이거나 감추는 기능을 수행하는 명령어가 CSON(커서보임) 과 CSOFF(커서감춤) 이다. LCD모듈에서 커서의 역할은 현재 문자열을 표시하고자 하는 위치를 가리키며, 일반적인 경우에는 커서를 감추고 사용한다. LCD모듈이 초기화되면 커서는 나타나지 않는 상태가 된다. 이 명령은 화면의 스크롤 기능의 ON/OFF로도 사용된다.

명령어 예

```
LCDINIT
CSON
PRINT "CURSOR ON"
```



CONT 글자밝기

LCD모듈에 나타나는 글자의 밝기를 조절한다.

명령어 구문

- 영문구문 : CONT *[밝기값]*
- 한글구문 : 글자밝기 *[밝기값]*

명령어 설명

미니로봇컨트롤러에 사용되는 16x2라인의 LCD모듈은 백라이트(Back Light)방식으로 사용시에는 액정 뒷면에 불이 켜진다. LCD모듈에 표시되는 글자은 검은 색으로 표시되는데, CONT(글자밝기) 명령어는 이 검은 색의 밝기를 조절한다. [밝기값]에는 숫자, 상수, 변수의 사용이 가능하다. [밝기값]이 작아질수록 검은 색이 어두어지므로 LCD모듈상에서는 점점 희미하게 나타난다. 초기 LCD모듈의 글자밝기 값은 7이다. 한글LCD모듈은 밝기조절기능은 없고 반전처리기능이 있다. 반전을 시킬 경우 밝기값을 0으로하면 반전된다.

명령어 예

```
LCDINIT

CONT 10
PRINT "miniROBOT"
```

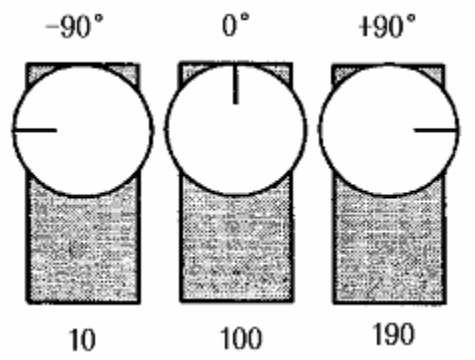
제 8 장

로보베이직

모터제어 명령어 설명

미니로봇 컨트롤러에서 제어할 수 있는 모터는 서보모터와 DC 모터가 있으며, DC모터의 제어는 일반적인 디지털 입출력 명령어를 사용하여 정지, 방향 제어를 할 수 있으며, PWM 관련 명령어를 사용하게 되면 속도 제어까지 가능하다.

서보모터는 -90°에서 +90°(동작범위 180°)의 범위 내에서 움직이므로, 로보베이직에서 모터를 동작시키기 위해서는 10에서 190까지의 수치로 표현한다. (미니로봇 컨트롤러에서는 음수 표현을 사용하지 않기 때문이다.)



그러므로, 서보모터의 기준각도인 0°가 되는 중심각도는 100이 된다.

여러 종류의 서보모터)

HS-311

HS-5645MG

HSR-8498HB

AIMOTOR

ZERO 영점

서보모터의 기준각도(영점)을 설정한다.

명령어 구문

MR-C2000 계열 컨트롤러)

- 영문구문 : ZERO *[모터0기준각도], [모터1기준각도], ..., [모터5기준각도]*
- 한글구문 : 영점 *[모터0기준각도], [모터1기준각도], ..., [모터5기준각도]*

MR-C3000 계열 컨트롤러)

- 영문구문 : ZERO *[그룹지정], [모터n기준각도], ...*
- 한글구문 : 영점 *[그룹지정], [모터n기준각도], ...*

명령어 설명

서보모터의 기준각도인 0°가 되는 중심각도 즉, 모터각도 100이 되는 영점(Zero Point)은 사용하는 서보모터마다 기어의 체결위치, 제품편차, 기타요인으로 오차가 발생하게 된다. 즉, 어떤 서보모터는 영점이 99 이거나 98일 수 있고, 또는 101이거나 102일 수 있다.

이러한 기준점의 오차는 서보모터를 동작시킬 때 또는 분해 조립 시 정확한 원위치를 잡는데 어려움을 준다. 이를 보정하기 위해서 ZERO(영점) 명령어를 사용한다. 서보모터를 사용하기 전에 미리 각 서보모터에 대한 영점을 설정하여두면, 동작(MOVE) 명령어를 사용할 때 설정된 영점각도를 기준으로 움직이게 된다.

MR-C2000 계열의 컨트롤러에서 영점 사용법)

영점을 잡는 작업은 초기에 한번만 하면 되고 명령이 실행되면 그 기준값은 EEPROM에 저장되어 보관되므로 전원 차단 후에도 계속 유지된다. 따라서 새로운 영점을 잡기 위해서는 다음의 단계를 거쳐야 한다. 로보베이직 최신 버전은 "영점설정"메뉴가 별도로 있어 이 과정이 필요없이 쉽게 할 수 있다.

1) 기존의 영점을 모두 100으로 설정한다. 즉 과거에 설정된 값들을 모두 제거한다.
2) 모터의 방향을 모두 정방향으로 설정한다.
3) 전원을 OFF한 후, 다시 ON 한다.
4) 온라인 기능으로 모터를 구동하여 중심점이라 생각되는 위치에 모터를 이동시킨다.
5) 이렇게 이동된 모터의 표시 값을 기억/기록 한 후 영점 명령으로 입력한다.

예1) 기존의 영점값을 제거한다.

```
ZERO 100,100,100,100,100,100
DIR   1,1,1,1,1,1                    '모터의 방향을 정방향으로 설정
```

예2) 영점을 재설정한다. 기록된 각 모터의 중심위치의 값을 기록한다.

```
DIR  1,1,1,1,1,1
ZERO 100, 101, 99                    '3개 서보모터의 영점설정
ZERO 102, 100, 100, 99, 101, 100     '6개 서보모터의 영점설정
```

MR-C2000 계열의 컨트롤러에서는 영점의 설정 각도를 90-110 사이의 값으로 사용하여야 한다.

MR-C3000 계열의 컨트롤러에서 영점 사용법)

MR-C3000에서 영점을 설정하고자 할 때는 반드시 그룹을 먼저 선언하여야 한다.

ZERO G8B, 80, 120, 115, 80, 117, 88, 95, 120

'그룹8B(서보모터 8번부터 15번까지) 각각 영점을 설정한다.

MR-C3000 계열 컨트롤러에서 영점설정은 80~120까지 설정 할 수 있다.

MOTOR 모터설정

서보모터의 사용여부를 설정한다.

명령어 구문

MR-C2000 계열의 컨트롤러)

- 영문구문 : MOTOR *[모터번호]*
- 한글구문 : 모터설정 *[모터번호]*

MR-C3000 계열의 컨트롤러)

- 영문구문 : MOTOR *[모터번호] / [그룹지정]*
- 한글구문 : 모터설정 *[모터번호] / [그룹지정]*

명령어 설명

MR-C2000 계열의 컨트롤러에서 사용설정)

MR-C2000 계열에서는 서보모터의 사용은 미니로봇컨트롤러의 0번 포트부터 5번 포트까지 6개 사용가능하며, *[모터번호]*는 0번 모터부터 5번 모터까지 가능하다.
모터번호가 6이면 1번부터 6번까지의 모든 모터에 대해 사용설정이 된다. 미니로봇컨트롤러에 모터를 연결하여 사용하고자 할 때는 반드시 프로그램의 처음 부분에 모터의 사용설정을 하여야 하며, *사용 설정없이 모터 동작에 관련된 명령어를 실행하게 되면 모터는 전혀 동작하지 않게 되므로* 항상 주의한다. *[모터번호]*에는 0에서 6까지의 숫자 또는 상수만 사용이 가능하다.

예1) MOTOR 6 *'모든 서보모터(0-5번) 사용설정*
예2) MOTOR 2 *'2번 서보모터 사용설정*

MR-C3000 계열의 컨트롤러에서 사용설정)

MR-C3000 계열에서 서보모터의 사용은 미니로봇 컨트롤러의 0번 포트부터 31번 포트까지 총 32개 사용가능하며, *[모터번호]*로 하나씩 모터를 지정하거나, *[그룹지정]*으로 한꺼번에 모터 사용설정이 가능하다.

*[모터번호]*는 숫자, 상수, 바이트형 변수 등이 모두 가능하므로, 다양한 사용설정 방법이 가능하다.

명령어 예

```
예1) MOTOR 0                 '서보모터 0번 모터를 사용 설정한다.
예2) MOTOR G6A               '서보모터 그룹6A(0~5번)을 사용 설정한다.
      MOTOR G6C              '서보모터 그룹6C(12~17번)을 사용 설정한다.
예3) MOTOR G8A               '서보모터 그룹8A(0~7번)을 사용 설정한다.
예4) 변수를 사용한 서보모터 사용설정
    DIM I AS BYTE
    FOR I = 0 TO 31          '변수 I를 사용해서 서보모터 0번부터 31번까지
                             '사용 설정합니다.
        MOTOR I
    NEXT I
예5) MOTOR G24               '서보모터 그룹24(0~23번)을 사용 설정한다.
예6) MOTOR ALLON             '모든 서보모터(0~31번)를 사용 설정한다.
```

MOTOROFF 모터해제

서보모터의 사용을 해제한다.

명령어 구문

MR-C2000 계열의 컨트롤러)

- 영문구문 : MOTOROFF *[모터번호]*
- 한글구문 : 모터해제 *[모터번호]*

MR-C3000 계열의 컨트롤러)

- 영문구문 : MOTOROFF *[모터번호] / [그룹지정]*
- 한글구문 : 모터해제 *[모터번호] / [그룹지정]*

명령어 설명

명령어의 내용은 모터설정(MOTOR) 명령어와 동일하며, 모터의 사용을 해제하게 되면 모터에 가해지는 힘(토크)가 없어지기 되며, 이후 사용되는 모터 동작 관련 명령어는 아무런 동작도 하지 않는다.

MOVE 동작

여러 개의 서보모터를 동시에 동작시킨다.

명령어 구문

MR-C2000 계열의 컨트롤러)

- 영문구문 : MOVE *[모터0각도], [모터1각도], ..., [모터5각도]*
- 한글구문 : 동작 *[모터0각도], [모터1각도], ..., [모터5각도]*

MR-C3000 계열의 컨트롤러)

- 영문구문 : MOVE *[그룹지정], [모터n각도], ...*
- 한글구문 : 동작 *[그룹지정], [모터n각도], ...*

명령어 설명

MR-C2000 계열의 컨트롤러에서 MOVE(동작) 명령어의 사용)

동작(MOVE)명령어는 서보모터를 원하는 각도로 동작시킨다. 이때, PWM 기능은 해제된다. ([주의] MOVE 명령과 PWM 명령은 한 프로그램에서 같이 사용 안 된다.) *[모터각도]*에는 10에서 190사이의 숫자나 상수를 기입한다. 사용모터가 1번과 3번, 4번일 경우에는

MOVE 60, , 100, 120

과 같이 2번 모터 자리는 생략한다. 또한, 2번 서보모터만 동작시킬 때는

MOVE , 140

과 같이 1번 모터와 3번부터 6번 모터까지의 동작각도를 생략하게 되면 해당 모터들은 이전의 각도를 계속 유지하고 2번 모터에 대해서만 140(+40°)만큼 동작한다.

MOVE(동작) 명령어에 대한 모터의 동작은 6개를 동시에 제어하기 때문에 로봇 등의 응용에서 움직임 제어가 쉽지는 않다. 또한, 모터들의 연결구조에 따라 모터각도가 의미하는 숫자는 아무런 정보도 제공하여 주지 않으므로, 미니로봇컨트롤러와 PC를 인터페이스 케이블로 연결한 후 로보베이직 상에서 *"서보모터 실시간 제어"*를 사용하면, 초보자도 쉽게 MOVE(동작) 명령어를 이용한 로봇 동작을 쉽게 작성할 수 있다.

서보모터를 개조하여 360°회전할 수 있게 한 경우에는 *[모터각도]*는 회전 속도를 의미한다. 각도가 100인 경우에는 모터가 멈추고, 100보다 작아질수록 좌회전(정회전)하는 속도가 빨라지며, 100보다 커질수록 우회전(역회전)하는 속도가 빨라진다. 그러므로, 개조된 서보모터에서는 MOVE(동작) 명령어로 회전의 방향과 정지, 회전 속도 등을 제어할 수 있다.

MOVE(동작) 명령어는 모터를 동작시키기 때문에 움직임의 시간이 요구된다. 그러므로, 실시간 OS가 내장된 미니로봇컨트롤러는 MOVE(동작) 명령어의 시작과 동시에 다음 명령어를 실행한다. 만약, 모터의 동작이 완료된 후에 다음 명령어를 실행하려면 WAIT(대기) 명령어를 사용한다.

명령어 예

```
MOVE 100, 50, 140, 120, 80, 40
MOVE 120, , , 160
MOVE , 70, 100
MOVE , , , , , 100
```

MR-C3000 계열의 컨트롤러에서 MOVE(동작) 명령어의 사용)

MR-C3000 계열 컨트롤러에서 서보모터 사용 포트와 PWM 사용 포트가 다르므로, MR-C2000 계열의 컨트롤러와 달리 동시에 사용이 가능하다.

명령어 예

예1)

```
MOVE   G6A, 85, 113, 72, 117, 115, 100
MOVE   G6C, 75,      , 96, 123,       , 122
MOVE   G8A, 85, 113, 72, 117, 115, 100, 95, 45
```

예2)

```
MOVE   G24, 85, 113, 72, 117, 115, 100
```

명령어의 사용과

```
MOVE24   85, 113, 72, 117, 115, 100
```

명령어는 같다.

SPEED 속도

서보 모터의 동작속도를 설정한다.

명령어 구문

- 영문구문 : SPEED *[모터속도]*
- 한글구문 : 속도 *[모터속도]*

명령어 설명

SPEED(속도) 명령어는 서보모터를 MOVE(동작) 명령어로 움직일 경우 동작속도를 설정한다. *MR-C2000 계열 컨트롤러에서는 [모터속도]에 1에서 15까지의 숫자 또는 상수를 사용할 수 있으며, MR-C3000 계열의 컨트롤러에서는 바이트형 변수의 사용이 가능하다.* 기본 모터동작 속도는 3이고, 숫자가 커질수록 동작속도는 빨라진다. 너무 빠른 동작속도를 사용하면 모터의 움직임이 빠르기 때문에 예기치 못한 사고가 발생할 수 있다. 그러므로, 적당한 모터동작 속도의 설정이 필요하다.

명령어 예

예1)

```
SPEED 7                    '모터속도를 7로 설정한다.
```

예2)

```
DIM STEP_SPEED AS BYTE     'STEP_SPEED라는 변수를 선언한다.
STEP_SPEED = 15            'STEP_SPEED라는 변수 값을 15로 설정한다.
SPEED STEP_SPEED           'SPEED 값을 STEP_SPEED이란 변수 값으로
                           '설정한다.
```

ACCEL 가속도

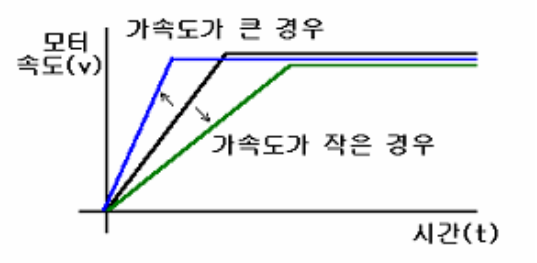
서보 모터의 동작 가속도를 설정한다. 2000

명령어 구문

- 영문구문 : ACCEL *[모터가속도]*
- 한글구문 : 가속도 *[모터가속도]*

명령어 설명

ACCEL(가속도) 명령어는 모터를 MOVE(동작) 명령어로 모터를 움직이려고 할 때 움직임 시작에서 일정속도로 유지되기까지의 속도변화 즉, 가속도를 설정한다.



*[모터가속도]*는 0에서 15까지의 숫자 또는 상수를 사용한다. 기본 모터동작 가속도는 3이고, 숫자가 커질수록 동작 가속도는 빨라지고, 움직임은 급격하게 동작한다.

모터 동작을 처음 시작할 경우, 현재 멈춰 있는 모터의 위치를 알 수 없기 때문에 바로 MOVE(동작) 명령어를 사용하면 갑작스럽게 모터들이 움직이게 된다. 이때 가속도와 속도를 작게 주면, 초기 MOVE(동작) 명령어에 대해서는 완만한 움직임이 가능하기 때문에 이러한 경우 ACCEL(가속도) 과 SPEED(속도) 명령어의 사용이 효과적이다.

MR-C3000 계열의 컨트롤러에서는 ACCEL(가속도) 명령어가 사용되지 않는다.

명령어 예

```
ACCEL 7       '서보모터 동작 가속도를 7로 설정한다.
```

DIR 방향

서보 모터의 동작 방향을 설정한다.

명령어 구문

MR-C2000 계열 컨트롤러)

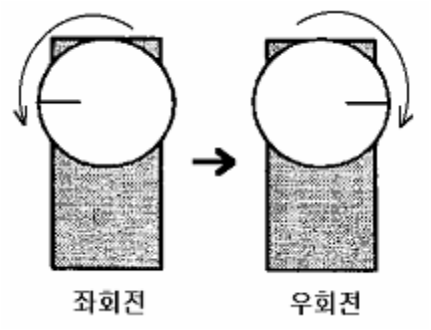
- 영문구문 : DIR *[모터0방향], [모터1방향], ..., [모터5방향]*
- 한글구문 : 방향 *[모터0방향], [모터1방향], ..., [모터5방향]*

MR-C3000 계열 컨트롤러)

- 영문구문 : DIR *[그룹지정], [모터n방향], ...*
- 한글구문 : 방향 *[그룹지정], [모터n방향], ...*

명령어 설명

서보모터는 기준각도인 100을 기준으로 작은 각도에 대해서는 좌회전, 큰 각도에 대해서는 우회전 한다. 아래 그림은 10이라는 각도에 대해서 정회전일 경우 좌회전을 하고, DIR(방향) 명령어에 의해 역회전을 할 경우에는 같은 10이라는 각도에 대해 우회전하는 서보모터를 나타내고 있다.



*[모터방향]*은 0 (역회전/좌회전) 또는 1 (정회전/우회전)의 숫자 또는 상수를 사용하며, 기본값으로는 1인 정회전 방향으로 설정되어 있다. 또한, MOVE(동작) 명령어에서와 마찬가지로 사용하고자 하는 모터에 대해서만 지정할 수 있다. 4번 까지 4개의 모터를 사용하는 경우 DIR 0, 1, , 0 과 같이 3번 모터방향을 생략하면 기본적인 정회전 1이 설정된다.

명령어 예

MR-C2000 계열의 컨트롤러 사용 예)

```
DIR 0, 1, 1, 0, 1, 0
DIR , , 0
```

MR-C3000 계열의 컨트롤러 사용 예)

```
DIR G8A, 0, 1, 0, 0, 1, 0, 0, 0
DIR G8B, 1, 0, 1, 1, 0, 1, 1, 1
```

PTP 점대점

서보 모터의 동시 동작 제어기능을 ON/OFF 설정한다.

명령어 구문

MR-C2000 계열 컨트롤러)

- 영문구문 : PTP *[설정값]*
- 한글구문 : 점대점 *[설정값]*

MR-C3000 계열 컨트롤러)

- 영문구문 : PTP *[SETON/SETOFF/ALLON/ALLOFF]*
- 한글구문 : 점대점 *[사용설정/사용해제/모두설정/모두해제]*

명령어 설명

서보모터를 여러 개 동작시킬 경우 움직이는 동작각도가 다를 경우 일정속도에서 모터들의 동작완료 시간이 다르다. 암로봇(Arm Robot)과 같이 여러 모터가 조합되어 움직이는 미니로봇 들의 경우에는 이러한 움직임이 매우 부자연스럽다. 로봇공학(Robot Engineering)에서는 모터들의 종료시점을 계산하여 동시에 동작 완료함으로써 움직임을 자연스럽게 연결하여 주는 점대점(Point to Point) 이론이 존재하는데, 미니로봇컨트롤러에는 점대점 이론의 움직임 제어기법이 내장되어 있다. PTP(점대점) 명령어는 이 움직임 제어기법을 가능하게 한다.

MR-C2000 계열 컨트롤러에서의 PTP(점대점) 제어)

*[설정값]*에는 0(해제) 또는 1(설정)의 숫자 또는 상수를 사용한다. 두 개의 서보모터를 1번 모터, 2번 모터로 사용할 경우, 다음의 프로그램의 예를 보자.

- 일반적인 부자연스러운 동작인 경우

```
PTP 0
MOVE 100, 100
MOVE 110, 120
```

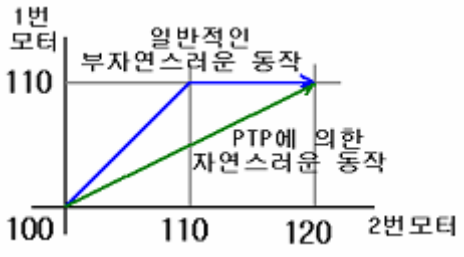
설명) 1번 모터는 10도 2번 모터는 20도를 움직이나 같은 속도로 움직이므로 1번과 2번이 10도를 같이 움직이고 2번은 나머지 10도를 혼자서 움직인다.

- PTP에 의한 자연스러운 동작인 경우

```
PTP 1
MOVE 100, 100
MOVE 110, 120
```

설명) 1번 모터는 10도 2번 모터는 20도를 움직이나 1번은 2번의 1/2속도로 움직이게 되어 있어서 1 ,2번 모터가 동시에 움직여서 1, 2번 모터가 동시에 정지한다.

PTP(점대점) 명령어를 사용한 경우와 사용하지 않는 경우의 모터 움직임에 대한 궤적을 그리면 다음과 같다.



즉, 일반적인 동작의 경우에는 모터1과 모터2가 110이라는 각도까지는 동시에 움직이지만, 110 이후에는 모터2만 120각도까지 움직이게 되어 부자연스러운 동작을 하게 된다.
하지만, PTP에 의한 동작의 경우에는 모터1과 모터2에 대해 움직일 각도 10과 20에 대해 PTP계산을 하고, 계산 값에 의해 동시에 완료하는 자연스러운 동작을 한다.

MR-C3000 계열 컨트롤러에서의 PTP(점대점) 제어)

MR-C3000의 경우 많은 모터가 사용되므로, 그룹별로 사용되는 MOVE 명령의 경우, 지정된 그룹에서만 PTP 기능을 수행할 것인지, 전체 모든 모터에 대해 PTP 기능을 수행할 것인지를 설정한다.

- PTP SETON(점대점 사용설정) : PTP 기능을 그룹별로 설정한다.
- PTP SETOFF(점대점 사용해제) : 그룹별로 설정된 PTP 기능을 해제한다.
- PTP ALLON(점대점 모두설정) : 모든 서보모터의 PTP기능을 설정한다.
- PTP ALLOFF(점대점 모두해제) : 모든 서보모터의 PTP기능을 해제한다.

MR-C3000 계열 컨트롤러에서 PTP 기능을 사용하는 경우 각 그룹동작간에 동작완료 지점에 *"WAIT"명령을 사용*하면 확실한 동작완료를 할 수 있다.

SERVO 서보

서보 모터를 1개 동작한다.

명령어 구문

- 영문구문 : SERVO *[모터번호],[모터각도]*
- 한글구문 : 서보 *[모터번호],[모터각도]*

명령어 설명

서보모터를 원하는 각도로 동작시킨다. MR-C2000 계열 컨트롤러의 경우 PWM 기능은 해제된다. *[모터번호]*는 서보모터가 연결된 모터 포트 번호이며, *[모터각도]*는 10-190 사이의 숫자, 상수 또는 바이트 형의 변수 사용이 가능하다.

명령어 예

예1)

```
SERVO 1,130                    '1번 모터를 130위치(각도 +30도)로 동작
```

예2)

```
DIM I AS BYTE

FOR  I = 10 TO 190
     SERVO 4, I
     DELAY 100
NEXT I
```

PWM 펄스폭

DC 모터의 속도를 제어하면서 동작시킨다.

명령어 구문

- 영문구문 : PWM *[모터번호],[펄스폭값]*
- 한글구문 : 펄스폭 *[모터번호],[펄스폭값]*

명령어 설명

MR-C2000 계열 컨트롤러에서 PWM(펄스폭) 제어)

MR-C2000 계열 컨트롤러는 서보모터 제어 포트와 PWM 포트를 공통으로 사용한다. 그러므로, *[모터번호]*는 0번부터 5번까지 사용한다. PWM(펄스폭) 명령은 모터 포트에 PWM 신호를 정해진 펄스 폭의 값으로 출력한다. 이때, 서보 기능은 해제된다. ([주의] SERVO/MOVE 명령과 PWM 명령은 한 프로그램에서 같이 사용 안 된다.)

듀티비	펄스 폭 값	듀티비	펄스 폭 값
0	0	60	153
10	25	70	178
20	51	80	204
30	77	90	229
40	102	100	254
50	127		

```
PWM 3, 127          '3번 모터 포트에 50% 듀티비의 PWM 출력
```

MR-C3000 계열 컨트롤러에서 PWM(펄스폭) 제어)

MR-C3000 계열 컨트롤러는 서보모터 제어 포트와 PWM 포트가 다르다. MR-C3000 계열에는 총 3개의 PWM 포트가 있으며, 각각 0번부터 2번까지 번호가 있다. 그러므로, *[모터번호]*는 0번부터 2번까지 사용한다. MR-C3000 계열의 PWM의 주파수는 454Hz이다.

```
PWM 0, 120          'PWM 0번 포트에 120 듀티비로 펄스를 발생한다.
```

FASTSERVO 빠른서보

서보 모터 1개를 빠르게 동작한다. 2000

명령어 구문

- 영문구문 : FASTSERVO *[모터번호],[모터각도]*
- 한글구문 : 빠른서보 *[모터번호],[모터각도]*

명령어 설명

서보모터를 원하는 각도로 가장 빠르게 동작시킨다. *MR-C2000 계열 컨트롤러에서만 사용*이 가능하다. *[모터번호]*는 서보모터가 연결된 모터 포트 번호이며, *[모터각도]*는 10-190 사이의 숫자, 상수만 가능하다.

명령어 예

FASTSERVO 2,190 *'2번 모터를 190까지 가장 빠르게 동작 시킨다.*

HIGHSPEED 고속동작

서보 모터의 동작을 고속모드로 설정한다. 3000

명령어 구문

- 영문구문 : HIGHSPEED *[SETON/SETOFF]*
- 한글구문 : 고속동작 *[사용설정/사용해제]*

명령어 설명

MR-C3000 계열 컨트롤러에서 서보모터의 동작을 고속 모드로 설정하거나 해제합니다. 현재 설정되어 있는 속도의 약 3배로 동작하게 된다.

- HIGHSPEED SETON (고속동작 사용설정) : MR-C3000 계열 컨트롤러에서 모든 서보모터를 가장 빠른 동작모드로 설정한다.
- HIGHSPEED SETOFF (고속동작 사용해제) : MR-C3000 계열 컨트롤러에서 모든 서보모터를 기본 동작속도로 동작한다.

명령어 예

HIGHSPEED SETON *'고속동작 모드를 설정한다.*

MOVEPOS 동작위치

POS 모터위치

모터의 특정 자세를 지정하고 동작시킨다. 3000

명령어 구문

- 영문구문 : MOVEPOS *[행라벨]*

.....

[행라벨]: POS *[그룹지정], [모터n각도],……*

- 한글구문 : 동작위치 *[행라벨]*

.....

[행라벨]: 모터위치 *[그룹지정], [모터n각도],……*

명령어 설명

MR-C3000 계열 컨트롤러에서 MOVE(동작)으로 이루어진 로봇의 자세를 다른 부분에서 참조할 경우, POS(모터위치) 명령어와 *[행라벨]*로 처리하고, 필요한 부분에서 MOVEPOS(동작위치) 명령어로 호출한다. MOVEPOS와 POS 명령어는 자주 반복하는 로보베이직의 프로그램을 간결하게 작성할 수 있고, 수정이 용이해진다.

명령어 예

```
.....
MOVEPOS  POS01          'POS01 라벨 위치의 POS 명령어 부분을
```

'동작시킨다.

.....

POS01: POS G6A, 10, 32, 15, 120, 78, 93

POS02: POS G6A, 67, 47, 32, 153, 23, 33

POS03: POS G6A, 34, 37, 122, 162, 84, 28

FPWM 주파수펄스

주파수 가변인 PWM 신호를 출력한다. 3000

명령어 구문

- 영문구문 : FPWM *[포트], [주파수], [튜티비]*
- 한글구문 : 주파수펄스 *[포트], [주파수], [튜티비]*

명령어 설명

MR-C3000 계열 컨트롤러에서 PWM의 주파수를 변경하여 펄스를 발생한다. 포트는 0~2까지, 주파수는 1(낮은 주파수)~5(높은 주파수)까지 사용하며, 튜티비는 0~255까지 사용 가능하다.

명령어 예

```
FPWM 0, 1, 127      'MR-C3000 계열 컨트롤러의 PWM 0번 포트에
                    '낮은 주파수로 50%의 듀티비(127) PWM 신호 출력
```

MOVE24 동작24

24개의 서보모터를 동작한다. 3000

명령어 구문

- 영문구문 : MOVE24 *[모터0각도],……,[모터23각도]*
- 한글구문 : 동작24 *[모터0각도],……,[모터23각도]*

명령어 설명

MR-C3000 계열 컨트롤러에서 사용되는 MOVE(동작) 명령어는 그룹지정을 통해 원하는 모터 그룹을 동작시킨다. MOVE24(동작24) 명령어는 “MOVE G24, ” 와 같이 동시에 24개의 서보모터를 동작시키는 명령어로, 16개 이상 24개 이하 서보 모터로 구성된 로봇을 동작시킬 때 효과적으로 사용된다.

명령어 예

MOVE24 100, 45, 67, 44, 132, 122, , , , 76, 81, 90

INIT 초기

로봇의 초기 동작 자세를 설정한다. 3000

명령어 구문

- 영문구문 : INIT *[그룹지정], [모터n각도],……*
- 한글구문 : 초기 *[그룹지정], [모터n각도],……*

명령어 설명

서보모터로 구성된 로봇에 MR-C3000 계열 컨트롤러를 장착하게 되면, 초기 전원 투입시 모든 서보모터가 100의 위치로 초기화된 후 MOVE(동작) 명령어에 의한 로봇 동작이 시작된다. 이러한 초기 동작이 전원 투입과 동시에 급격한 모터의 동작을 발생시키기 때문에 로봇의 파손의 위험이 종종 일어난다. 이를 방지하기 위하여 초기 자세를 INIT(초기) 명령어로 지정하여 전원 투입시 초기 100위치가 아닌 INIT로 지정된 모터의 위치로 이동시켜 로봇의 급격한 동작을 방지할 수 있다.

INIT(초기) 명령은 GETMOTORSET(모터입력설정) 명령과 유사한 기능을 수행하며, 일반적인 경우 아날로그 서보모터 (HITEC사의 HS- 시리즈)의 경우 INIT 명령을, 로봇 전용의 디지털 서보모터 (HITEC사의 HSR- 시리즈)의 경우 GETMOTORSET 명령을 사용한다.

명령어 예

INIT G8A, 100, 45, 67, 44, 132, 122, 76, 81

MOTORIN() 모터입력()

서보모터의 현재 값을 읽어 들인다. 3000

명령어 구문

- 영문구문 : MOTORIN (*[모터번호]*)
- 한글구문 : 모터입력 (*[모터번호]*)

명령어 설명

MR-C3000 계열 컨트롤러에 연결된 로봇 전용 서보모터 (HITEC사의 HSR- 시리즈)의 현재 서보모터 위치 값을 입력 받아, 프로그램으로 쉽게 제어할 수 있게 한다.

*[모터번호]*는 0-31까지의 숫자, 상수가 가능하며, 서보모터가 연결된 경우 10-190의 모터 각도를 읽어 들이게 되고, 서보모터가 연결되어 있지 않는 경우에는 0 값을 읽어 들이게 된다.

명령어 예

```
DIM  S0  AS  BYTE

MOTOR 0                 '0번 서보모터를 사용한다.
S0 = MOTORIN(0)         '0번 서보모터의 값을 S0 변수에 저장한다.
```

AIMOTOR AI모터설정

AI서보모터의 사용여부를 설정한다. 3000

명령어 구문

- 영문구문 : AIMOTOR *SETON/SETOFF/INIT/[모터번호] / [그룹지정]*
- 한글구문 : AI모터설정 *사용설정/사용해제/초기/[모터번호] / [그룹지정]*

명령어 설명

AI모터는 메가로보틱스(Megarobotics)사에서 개발한 로봇 전용의 새로운 서보모터이다. AI모터의 내부에 마이크로 컨틀로러 칩이 내장되어 RS232통신 규격으로 미니로봇 컨트롤러와 통신하게 되며, 모터의 각도를 제어할 수 있을 뿐만 아니라 현재 모터의 상태와 PID제어 계수(PGAIN, DGAIN)를 통한 모터 힘(토크)의 제어까지도 가능하다. AI모터는 미니로봇 컨트롤러에서 서보모터와 같은 개념으로 제어할 수 있다.

*AI모터의 사용은 미니로봇 컨트롤러의 0번 포트부터 30번 포트까지 총 **31개** 사용*가능하며, *[모터번호]*로 하나씩 모터를 지정하거나, *[그룹지정]*으로 한꺼번에 모터 사용설정이 가능하다. 명령어를 사용하는 방법은 MOTOR(모터설정) 명령어와 같다. *[모터번호]*는 숫자, 상수, 바이트형 변수 등이 모두 가능하다.

AI모터는 일반 서보모터와 달리 사용하기 전에 반드시 “AI모터를 사용한다”라는 선언을 해주어야 한다.

- AIMOTOR SETON (AI모터설정 사용설정) : AI모터의 사용을 가능하게 설정한다.
- AIMOTOR SETOFF (AI모터설정 사용해제) : AI모터의 사용을 해제한다.
- AIMOTOR INIT (AI모터설정 초기) : AI모터를 초기 위치로 부드럽게 동작한다.

명령어 예

```
AIMOTOR INIT        'AI모터를 초기화
AIMOTOR SETON       'AI 모터를 앞으로 사용할 것이라는 선언
AIMOTOR 0           '0번 AIMOTOR를 사용한다.
AIMOTOR G6B         '그룹6B 모터(6번-11번)를 모두 사용한다.
```

AIMOTOROFF AI모터해제

AI모터의 사용을 해제한다. 3000

명령어 구문

- 영문구문 : AIMOTOROFF *[모터번호] / [그룹지정]*
- 한글구문 : AI모터해제 *[모터번호] / [그룹지정]*

명령어 설명

AIMOTOR(AI모터설정) 명령어로 설정된 AI모터의 사용 가능 설정을 해제한다. MOTOROFF(모터해제) 명령과 같은 기능을 수행한다.

명령어 예

```
AIMOTOROFF 0          '0번 AIMOTOR를 해제한다.
AIMOTOROFF G6B        '그룹6B 모터(6번-11번)를 모두 해제한다.
AIMOTOR SETOFF        'AI 모터를 앞으로 사용하지 않을 것이라는 선언
```

AIMOTORIN() AI모터입력()

AI모터의 현재 값을 읽어들인다. 3000

명령어 구문

- 영문구문 : AIMOTORIN (*[모터번호]*)
- 한글구문 : AI모터입력 (*[모터번호]*)

명령어 설명

MR-C3000 계열 컨트롤러에 연결된 AI모터의 현재 위치 값을 입력 받아, 프로그램으로 쉽게 제어할 수 있게 한다.

*[모터번호]*는 0-30까지의 숫자, 상수가 가능하며, 서보모터가 연결된 경우 10-190의 모터 각도를 읽어 들이게 되고, 서보모터가 연결되어 있지 않는 경우에는 0 값을 읽어 들이게 된다.

명령어 예

```
DIM  AI5  AS  BYTE

AIMOTOR INIT
AIMOTOR SETON
AIMOTOR 5                 '5번 AI모터 사용설정
AI5 = AIMOTORIN(5)        '5번 AI모터의 값을 AI5 변수에 저장한다.
```

GETMOTORSET 모터입력설정

현재 서보모터의 값을 읽어 들여 현재 상태를 유지한다. 3000

명령어 구문

- 영문구문 : GETMOTORSET *[그룹지정], [모터n입력설정지정],…*
- 한글구문 : 모터입력설정 *[그룹지정], [모터n입력설정지정],…*

명령어 설명

MR-C3000 계열 컨트롤러는 디지털 로봇 전용 서보모터인 HSR- 시리즈를 연결하여 사용할 경우, 서보모터와의 통신을 통해 현재 서보모터의 값을 읽어 들일 수 있다. 로봇에 처음 전원을 인가하게 되면, 컨트롤러 초기화로 인해 모든 서보모터의 초기값인 100 위치로 이동하기 때문에 로봇의 동작이 급격하게 발생되며, 이를 방지하기 위하여 현재 모터 값을 읽어 이를 서보모터의 초기값으로 설정하게 되면, 로봇은 그대로 자세를 유지한 후, MOVE(동작) 명령어에 의한 로봇동작이 시작되기 때문에 자연스러운 초기 동작을 보이게 된다.

[모터n입력설정지정] 에는 “0” 또는 “1” 을 사용하며, “1”인 경우에는 지정된 서보모터의 현재 값을 읽어 들여 현재 로봇의 상태를 유지하게 되며, “0’인 경우에는 컨트롤러의 초기값인 100위치로 이동한다.

명령어 예

GETMOTORSET G8A, 1, 1, 1, 1, 0, 0, 0, 0

‘0번, 1번, 2번, 3번 서보모터는 현재 서보모터의 값을 전원 투입시 그대로 유지한다.

‘4번, 5번, 6번, 7번 서보모터는 전원 투입시 모두 초기 100위치로 이동한다.

OFFSET 옵셋

서보모터의 기준각도(영점)을 변경한다. 3000

명령어 구문

MR-C3000 계열 컨트롤러)

- 영문구문 : OFFSET *[그룹지정], [모터n영점설정각도], ...*
- 한글구문 : 옵셋 *[그룹지정], [모터n영점설정각도], ...*

명령어 설명

MR-C3000 계열 컨트롤러를 사용한 로봇은 조립하는 모터의 위치에 따라 움직이는 각도의 값이 조금씩 달라지기 때문에 “ZERO(영점)” 명령어를 사용한다. 이 ZERO 명령어는 컨트롤러에 탑재된 플래시 메모리에 기억되어 있으며, 초기 전원이 투입될 때, 한 번 참조되어, 모터의 동작에 반영한다. “OFFSET(옵셋)” 명령어는 ZERO 명령어와 달리 메모리에 기억되지 않고, 바로 프로그램 내에서 영점의 변경을 원할 경우에 사용된다.
영점 설정 각도를 기입하지 않으면, 기존의 영점 값을 유지한다.

명령어 예

OFFSET G6A, 100, 98, , 102, 100, 106

제 9 장

로보베이직

음악제어 명령어 설명

미니로봇컨트롤러를 이용해서 만든 응용시스템에 소리를 내는 효과는 주변 사람들의 효과를 끌만큼 크다. 미니로봇컨트롤러에서는 간단한 음향효과를 낼 수 있을 뿐만 아니라 동요 등의 간단한 악보를 보면서 프로그램하면 음악을 연주할 수 있는 기능을 제공한다. 미니로봇컨트롤러를 사용해서 소리를 내기 위해서는 별도의 부품이 필요한데 주로 구조가 간단한 피에조(PIEZO)를 사용한다. 피에조는 + 전원단자(빨간색)와 신호단자(흰색)로 구성되어 있다.

좀 더 크고, 깨끗한 소리를 듣기 위해서는 스피커에 연결하는데, 이때는 전류 드라이버 회로나 증폭회로(AMP)가 필요하다.

MR-C2000 계열의 컨트롤러에 피에조를 연결하는 예)

MR-C2000 계열 컨트롤러의 8번 포트에 피에조를 연결한다. 피에조의 + 단자는 8번 포트의 VCC 단자에, - 단자는 8번 포트의 SIG(신호) 단자에 연결한다.

MR-C3000 계열의 컨트롤러에 피에조를 연결하는 예)

MR-C3000 계열 컨트롤러의 28번 포트에 피에조를 연결한다. 피에조의 + 단자는 28번 포트의 VCC 단자에, - 단자는 28번 포트의 SIG(신호) 단자에 연결한다.

BEEP 삑

피에조에 경고음(“삑”)을 낸다. 2000

명령어 구문

- 영문구문 : BEEP
- 한글구문 : 삑

명령어 설명

MR-C2000 계열 컨트롤러에서 간단한 경고음(“삑”)을 발생하고자 할 때 BEEP(삑) 명령어를 사용한다. 보통 경고음만 발생시키기 위해서는 피에조 대신 부저(Buzzer)를 많이 사용하며, 부저는 일반 출력 포트에 연결하여 OUT(출력) 명령어로 간단히 소리를 낼 수 있다. *MR-C3000 계열 컨트롤러에서는 부저와 OUT(출력) 명령을 사용한다.*

명령어 예

```
BEEP                ‘경고음 발생
```

SOUND 소리

피에조에 주파수 음을 발생한다. 2000

명령어 구문

- 영문구문 : SOUND *[음정], [길이], [음정], [길이], ...*
- 한글구문 : 소리 *[음정], [길이], [음정], [길이], ...*

명령어 설명

MR-C2000 계열 컨트롤러에서 피에조 포트에 원하는 주파수의 신호와 지연시간을 직접 설정할 수 있다. 설정 가능한 값은 1부터 254 까지이고 각 값에 일치하는 주파수는 아래 표와 같다. 길이는 단위가 약 11msec 이다.

입력값	주파수(Hz)	입력	주파수(Hz)	입력	주파수(Hz)
1	38.86k	70	800	160	389
2	23.81k	80	775	170	365
5	11.11k	90	689	180	344
10	5.88k	100	621	190	327
20	3.00k	110	565	200	311
30	2.00k	120	518	210	295
40	1.54k	130	478	220	283
50	1.23k	140	444	230	270
60	1.00k	150	413	240	260

시간	길이
0.5초	45
1초	90
2초	180

명령어 예

SOUND 60, 90, 60, 180, 30, 45

'1KHz 주파수를 1초간, 2초간 발생하고 2KHZ 주파수를 0.5초 발생한다.

PLAY 연주

피에조에 노래를 연주한다. 2000

명령어 구문

- 영문구문 : PLAY "*[연주문자열]*"
- 한글구문 : 연주 "*[연주문자열]*"

명령어 설명

로보베이직과 미니로봇 컨트롤러는 음악을 연주할 수 있는 기능을 제공한다. *[연주문자열]*은 연주데이터가 들어가는데, 문자열에 사용할 수 있는 문자와 의미는 다음과 같다.

문자열		설 명
영어/기호	한글	
C	도	계명 “도”
D	레	계명 “레”
E	미	계명 “미”
F	파	계명 “파”
G	솔	계명 “솔”
A	라	계명 “라”
B	시	계명 “시”
T	템포, 박자, 속도	연주 박자(속도)를 조절한다.
L	저	저음 옥타브를 선택한다.
M	중	중음 옥타브를 선택한다.
H	고	고음 옥타브를 선택한다.

#, +		반음 올린다.(#)
$, -		반음 내린다.(♭)
P, ,(쉼표)	쉼	소리를 내지 않고 일정길이만큼 쉰다.
<		한 옥타브 내린다.
>		한 옥타브 올린다.

템포(T)는 연주할 때 음의 빠르기를 설정하며, 기본 템포 값은 7이고, 1에서 0까지 설정이 가능하다. 템포 값 0은 10을 의미하며, 가장 느린 빠르기를 갖고, 템포 값 1은 가장 빠른 빠르기를 갖는다.

MR-C2000 계열 컨트롤러에서 처리할 수 있는 옥타브(Octave)는 3단계이며, 각각 저, 중, 고 옥타브라 불린다.

음 길이는 0-9까지의 숫자를 사용하며, 숫자의 의미는 다음과 같다.

음표	온음표	2분 음표	점2분 음표	4분 음표	점4분 음표	8분 음표	점8분 음표	16분 음표	점16분 음표	32분 음표
	𝅝	𝅗𝅥	𝅗𝅥.	♩	♩.	♪	♪.	𝅘𝅥𝅯	𝅘𝅥𝅯.	𝅘𝅥𝅰
숫자	1	2	3	4	5	8	9	6	7	0

음 길이를 생략하면 바로 그 전의 음 길이로 연주한다. "4CDEF8G" 는 4분 음표 도, 레, 미, 파, 8분 음표의 솔을 연주한다. PLAY(연주) 문이 사용되면 기본적으로 중음 옥타브에, 4분 음표, 계명은 도, 템포 7로 설정되고, 다음 PLAY(연주)에서는 이전의 옥타브와 템포를 이어받는다.

① 음의 길이는 음의 앞에 붙인다. [4도 8미 6파]
② #이나 플렛은 음의 앞에 붙인다. [#도 $미 +파 -솔 4#도 #4도]

MR-C3000 계열의 컨트롤러에서는 PLAY(연주) 명령어 대신 MUSIC(음악) 명령어를 사용한다.

명령어 예

```
PLAY “M4GGAA GGE GGEED”
PLAY “M4GGAA GGE GEDEC”
```

MUSIC 음악

피에조에 음악를 연주한다. 3000

명령어 구문

- 영문구문 : MUSIC "*[연주문자열]*"
- 한글구문 : 음악 "*[연주문자열]*"

명령어 설명

로보베이직과 미니로봇 컨트롤러는 음악을 연주할 수 있는 기능을 제공한다. *[연주문자열]*은 연주데이터가 들어가는데, 문자열에 사용할 수 있는 문자와 의미는 다음과 같다.

문자열		설 명
영어/기호	한글	
C	도	계명 “도”
D	레	계명 “레”
E	미	계명 “미”
F	파	계명 “파”
G	솔	계명 “솔”
A	라	계명 “라”
B	시	계명 “시”
[		잇단음표를 위해 음의 길이를 1.5배 짧게
]		잇단음표를 위해 음의 길이를 1.5배 길게
O	옥	옥타브를 선택한다.
M	중	3 옥타브를 선택한다.

.		음을 1.5배 길게 한다.
#, +		반음 올린다.(#)
$, -		반음 내린다.(♭)
P, ,(쉼표)	쉼	소리를 내지 않고 일정길이만큼 쉰다.
<, L		한 옥타브 내린다.
>, H		한 옥타브 올린다.

MR-C3000 계열 컨트롤러에서 처리할 수 있는 옥타브(Octave)는 7단계이며, 음 길이는 0-9까지의 숫자를 사용하며, 숫자의 의미는 다음과 같다. 점이 붙은 음표의 경우 숫자로 표시하거나, *[연주문자열]* 중에 "."으로 표시할 수 있다.

음표	온음표	2분 음표	점2분 음표	4분 음표	점4분 음표	8분 음표	점8분 음표	16분 음표	점16분 음표	32분 음표
	𝅝	𝅗𝅥	𝅗𝅥.	♩	♩.	♪	♪.	𝅘𝅥𝅯	𝅘𝅥𝅯.	𝅘𝅥𝅰
숫자	1	2	3	4	5	8	9	6	7	0

MR-C2000 계열의 컨트롤러에서는 MUSIC(음악) 명령어 대신 PLAY(연주) 명령어를 사용한다.

MR-C3000 계열의 컨트롤러는 박자(Tempo)를 위해서 TEMPO(박자) 명령어가 별도로 있으므로, 해당 명령어를 참조한다.

명령어 예

MUSIC “O34GGAA GGE GGEED”
MUSIC “O3GGA4.A GGE GEDEC”

TEMPO 박자

음악을 연주할 박자를 설정한다. 3000

명령어 구문

- 영문구문 : TEMPO *[설정값]*
- 한글구문 : 박자 *[설정값]*

명령어 설명

MR-C3000 계열 컨트롤러에서 MUSIC(음악) 명령어를 사용하여 연주할 음악의 박자를 설정한다.

*[설정값]*은 0-255 사이의 숫자, 상수, 바이트형 변수를 사용할 수 있다.

제 10 장

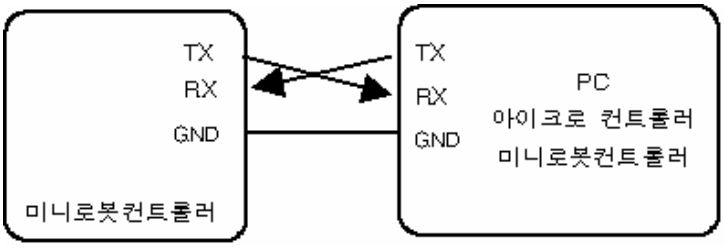
로보베이직

외부통신 명령어 설명

MR-C2000 계열 컨트롤러의 외부통신)

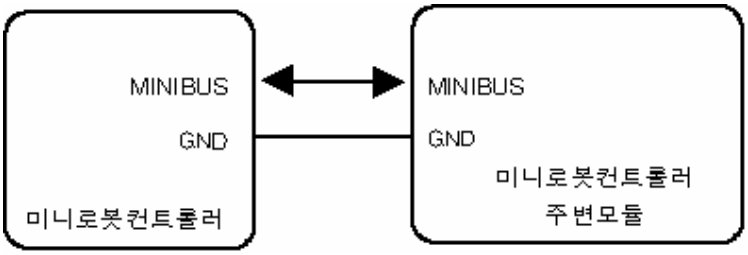
MR-C2000 계열 컨트롤러는 외부와의 통신을 위하여 두 가지 방식의 방법을 사용할 수 있다. 하나는 RS232방식의 시리얼 통신(Serial Communication)이고, 다른 하나는 자체 미니버스(miniBUS)를 사용하는 방법이다.

먼저, RS232방식의 시리얼 통신은 RS232를 지원하는 일반 PC나 다른 마이크로 컨트롤러(다른 미니로봇컨트롤러도 포함)와 데이터를 주고 받을 수 있다. 또한, 유선 또는 무선(RF모듈 사용)으로 데이터를 주고 받을 수 있으므로 많은 장점이 있다. RS232통신을 하기 위해서는 3선이 필요한데, 송신(TX)와 수신(RX) 그리고 접지(GND)를 서로 연결하여 주면 된다.



참고로, 일반 PC와 연결하기 위해서는 부가적인 MAX232와 같은 전압변환용 회로가 필요하다.

미니버스방식은 RS232통신이 3선을 사용하는데 반하여, BUS신호선과 접지(GND)의 2선으로 데이터를 주고받는다.



미니버스는 한 선으로 데이터를 보내고 받으므로, 컨트롤러들 간에 서로 정확한 규칙에 의해 데이터를 주고받아야 제대로 된 데이터전송을 할 수 있다. 미니버스에 의한 데

이터 전송의 예는 LCD모듈을 들 수 있으며, 여기서 LCD모듈에 장착된 제어컨트롤러는 미니버스를 수신전용으로 사용하고 있다.

MR-C3000 계열 컨트롤러의 외부통신)

MR-C3000 계열 컨트롤러에는 외부 기기 또는 소자들과 고속으로 RS232통신할 수 있는 기능이 포함되어 있다. 단, MR-C3000 계열에서는 미니버스(miniBUS) 통신은 지원하지 않는다. RS232 통신은 MR-C2000과 같은 연결로 가능하며, 최대 115,200bps 속도까지 가능하다.

RX 수신

RX포트를 이용하여 RS232 신호를 수신한다. 2000

명령어 구문

- 영문구문 : RX *[포트속도], [수신변수], [수신에러처리라벨]*
- 한글구문 : 수신 *[포트속도], [수신변수], [수신에러처리라벨]*

명령어 설명

MR-C2000 계열 컨트롤러의 9번 포트를 이용하여 RS232방식의 데이터를 수신한다. *[포트속도]*는 1~4까지의 숫자나 상수를 사용하며, *[포트속도]*의 의미는 다음과 같다.

숫자	포트 설정
1	1200bps, 8Bit data, No parity, 1 Stop bit
2	2400bps, 8Bit data, No parity, 1 Stop bit
3	2400bps, 8Bit data, No parity, 1 Stop bit
4	4800bps, 8Bit data, No parity, 1 Stop bit

*[수신변수]*는 수신된 데이터를 저장할 변수를 말하며, 반드시 선언된 바이트형 변수를 사용한다.

*[수신에러처리라벨]*은 데이터가 수신되지 않았으면(통신버퍼가 비어있으면), 처리하는 부분이 시작되는 위치의 라벨을 말한다. RS232포트를 통해서 데이터가 수신될 때까지 기다리는 프로그램은 다음과 같이 작성할 수 있다.

```
Retry:
        RX  4,  A,  Retry
```

MR-C3000 계열 컨트롤러에서 RS232 신호 수신을 위해서는 RX(수신) 명령어 대신 ERX(데이타수신) 명령어를 사용한다.

명령어 예

다음 예는 외부에 연결된 RS232 단자를 통해 아스키 코드 &h80(16진수) 값이 들어오면 0번 포트의 LED를 켜고, 그 외에는 LED를 끄는 프로그램 예이다.

```
        DIM   A   AS   BYTE

Retry:  RX 4, A, Retry
        IF A = &h80 THEN
                OUT 0, 0
        ELSE
                OUT 0, 1
        ENDIF
        GOTO Retry
```

TX 송신

TX포트를 이용하여 RS232 신호를 송신한다. 2000

명령어 구문

- 영문구문 : TX *[포트속도], [데이터]*
- 한글구문 : 송신 *[포트속도], [데이터]*

명령어 설명

MR-C2000 계열 컨트롤러의 10번 포트를 이용하여 RS232방식의 데이터를 송신한다. *[포트속도]*는 1~4까지의 숫자나 상수를 사용하며, *[포트속도]*의 의미는 다음과 같다.

숫자	포트 설정
1	1200bps, 8Bit data, No parity, 1 Stop bit
2	2400bps, 8Bit data, No parity, 1 Stop bit
3	2400bps, 8Bit data, No parity, 1 Stop bit
4	4800bps, 8Bit data, No parity, 1 Stop bit

*[데이터]*는 송신포트를 통해 보내는 데이터 값이다. *[데이터]*는 숫자, 상수, 변수 등을 사용할 수 있다. 만약 "A"문자를 송신하려고 하면, "A"에 해당하는 아스키 코드를 보내야 하는데, 다음과 같이 하면 간단히 송신할 수 있다.

```
DIM I AS BYTE

I = "A"
```

```
TX 4, I
```

즉, 문자열을 변수에 대입하게 되면 문자열 중 맨 첫 글자의 아스키 코드 값이 변수에 대입된다.

MR-C3000 계열 컨트롤러에서 RS232 신호 송신을 위해서는 TX(송신) 명령어 대신 ETX(데이타송신) 명령어를 사용한다.

명령어 예

다음 예는 외부에 연결된 RS232 단자를 통해 바이트포트 0번의 키 값을 계속 송신하는 프로그램의 예이다.

```
DIM  A  AS  BYTE

Main:
    A = BYTEIN(0)
    TX  4, A
    GOTO Main
```

MINIIN 미니입력

미니버스 포트를 이용하여 미니버스신호를 수신한다. 2000

명령어 구문

- 영문구문 : MINIIN
- 한글구문 : 미니입력

명령어 설명

MR-C2000 계열 컨트롤러의 미니버스 포트(6번 포트)를 이용하여 미니버스방식의 데이터를 수신한다. 만약 수신된 데이터가 없으면 0값이 들어온다.
미니버스를 통하여 데이터가 들어올 때까지 대기하는 프로그램은 다음과 같이 작성하면 된다.

```
        DIM A AS BYTE

Retry:
        A = MINIIN
        IF A = 0 THEN GOTO Retry
```

MINIOUT 미니출력

미니버스 포트를 이용하여 미니버스신호를 송신한다. 2000

명령어 구문

- 영문구문 : MINIOUT *[데이터] , [데이터], ...*
- 한글구문 : 미니출력 *[데이터] , [데이터], ...*

명령어 설명

MR-C2000 계열 컨트롤러의 미니버스 포트(6번 포트)를 이용하여 미니버스방식의 데이터를 송신한다. 미니버스는 RS232방식의 “4800bps, 8Bit data, No parity, 1 Stop bit”의 포트 설정과 같은 기능을 수행한다. *[데이터]*는 숫자, 상수, 변수 등을 사용할 수 있으며, 무한개의 데이터를 보낼 수 있다. 단, *0이라는 숫자는 보낼 수 없다*는 것에 주의한다.

명령어 예

MINIOUT 100, 20, 76, 65

ERX 데이타수신

ERX포트를 이용하여 RS232 신호를 수신한다. 3000

명령어 구문

- 영문구문 : ERX *[포트속도], [수신변수], [수신에러처리라벨]*
- 한글구문 : 데이타수신 *[포트속도], [수신변수], [수신에러처리라벨]*

명령어 설명

MR-C3000 계열 컨트롤러의 ERX 포트를 이용하여 RS232방식의 데이터를 수신한다. *[포트속도]* 의 의미는 다음과 같다.

숫자	포트 설정
2400	2400bps, 8Bit data, No parity, 1 Stop bit
4800	4800bps, 8Bit data, No parity, 1 Stop bit
9600	9600bps, 8Bit data, No parity, 1 Stop bit
14400	14400bps, 8Bit data, No parity, 1 Stop bit
19200	19200bps, 8Bit data, No parity, 1 Stop bit
28800	28800bps, 8Bit data, No parity, 1 Stop bit
38400	38400bps, 8Bit data, No parity, 1 Stop bit
57600	57600bps, 8Bit data, No parity, 1 Stop bit
76800	76800bps, 8Bit data, No parity, 1 Stop bit
115200	115200bps, 8Bit data, No parity, 1 Stop bit
230400	230400bps, 8Bit data, No parity, 1 Stop bit

*[수신변수]*는 수신된 데이터를 저장할 변수를 말하며, 반드시 선언된 바이트형 변수를 사용한다.

*[수신에러처리라벨]*은 데이터가 수신되지 않았으면(통신버퍼가 비어있으면), 처리하는 부분이 시작되는 위치의 라벨을 말한다. RS232포트를 통해서 데이터가 수신될 때까지 기다리는 프로그램은 다음과 같이 작성할 수 있다.

```
Retry:
        ERX  9600,  A,  Retry
```

MR-C2000 계열 컨트롤러에서 RS232 신호 수신을 위해서는 ERX(데이타수신) 명령어 대신 RX(수신) 명령어를 사용한다.

ETX 데이타송신

ETX포트를 이용하여 RS232 신호를 송신한다. 3000

명령어 구문

- 영문구문 : ETX *[포트속도], [데이터]*
- 한글구문 : 데이타송신 *[포트속도], [데이터]*

명령어 설명

MR-C3000 계열 컨트롤러의 ETX 포트를 이용하여 RS232방식의 데이터를 송신한다. *[포트속도]* 의 의미는 다음과 같다.

숫자	포트 설정
2400	2400bps, 8Bit data, No parity, 1 Stop bit
4800	4800bps, 8Bit data, No parity, 1 Stop bit
9600	9600bps, 8Bit data, No parity, 1 Stop bit
14400	14400bps, 8Bit data, No parity, 1 Stop bit
19200	19200bps, 8Bit data, No parity, 1 Stop bit
28800	28800bps, 8Bit data, No parity, 1 Stop bit
38400	38400bps, 8Bit data, No parity, 1 Stop bit
57600	57600bps, 8Bit data, No parity, 1 Stop bit
76800	76800bps, 8Bit data, No parity, 1 Stop bit
115200	115200bps, 8Bit data, No parity, 1 Stop bit
230400	230400bps, 8Bit data, No parity, 1 Stop bit

*[데이터]*는 송신포트를 통해 보내는 데이터 값이다. *[데이터]*는 숫자, 상수, 변수 등을 사용할 수 있다. 만약 "A"문자를 송신하려고 하면, "A"에 해당하는 아스키 코드를 보내야 하는데, 다음과 같이 하면 간단히 송신할 수 있다.

```
DIM I AS BYTE

I = "A"
ETX  9600, I
```

즉, 문자열을 변수에 대입하게 되면 문자열 중 맨 첫 글자의 아스키 코드 값이 변수에 대입된다.

MR-C2000 계열 컨트롤러에서 RS232 신호 송신을 위해서는 ETX(데이타송신) 명령어 대신 TX(송신) 명령어를 사용한다.

제 11 장

로보베이직

아날로그 신호처리

명령어 설명

AD() 변환()

AD변환포트로부터 아날로그 신호를 디지털로 변환한다. 3000

명령어 구문

- 영문구문 : AD (*[AD포트]*)
- 한글구문 : 변환 (*[AD포트]*)

명령어 설명

MR-C3000 계열 컨트롤러에는 0번부터 7번까지의 총 8개의 AD변환 포트 (디지털입출력 포트 32번부터 39번까지) 가 있으며, 외부 센서 또는 기기로부터의 아날로그 신호를 디지털로 변환하여 읽어 들인다. *[AD포트]* 에는 0부터 7까지 숫자, 상수 또는 바이트형 변수를 사용한다.

명령어 예

다음 예제는 AD 1번 포트로부터 아날로그 신호를 입력 받아 그 값을 LCD 모듈에 출력하는 예제이다.

```
        DIM a AS BYTE       'a라는 바이트형 변수를 선언한다.
        LCDINIT             'LCD모듈 사용을 초기화한다.
        CLS                 'LCD화면을 모두 지운다.
        CSOFF               '커서를 보이지 않게 한다.
MAIN:                       'MAIN이라는 라벨을 선언한다.
        a = AD(1)           'AD 1번 포트에서 입력된 값을 a라는 변수에
```

```
                        '저장한다.
LOCATE 5,0              '커서를 5,0 에 위치 시킨다.
PRINT FORMAT(a,DEC,2)   '입력된 a값을 십진법으로 2자리수 만큼
                        'LCD 모듈에 출력한다.
GOTO MAIN               'MAIN 이라는 라벨로 분기한다.
```

REMOCON() 리모콘()

AD변환포트 7번으로부터 적외선 리모콘의 값을 읽어온다. 3000

명령어 구문

- 영문구문 : REMOCON (*[리모콘 종류]*)
- 한글구문 : 리모콘 (*[리모콘종류]*)

명령어 설명

MR-C3000 계열 컨트롤러의 AD 변환포트 7번 포트를 이용하여 적외선 리모콘의 눌린 키의 값을 읽어온다. *[리모콘 종류]* 에는 0번부터 정해진 리모콘 종류에 따라 사용하며, 사용할 수 있는 리모콘의 종류는 MR-C3000 계열 컨트롤러의 버전에 따라 추가될 수 있다. 자세한 내용은 뒤 부분의 적외선 리모콘을 사용한 예제를 참조한다.

리모콘 종류 0번 (모델명 : MR-IR2)

명령어 예

```
DIM a AS BYTE                    '수신된 값을 받을 변수
MAIN:                            '계속 리모콘의 값을 입력받기 위한 MAIN 라벨
```

```
a = REMOCON(0)                    '리모콘에서 수신된 값을 a라는 변수에 넣는다.
ON a GOTO MAIN,KEY1,KEY2,KEY3,KEY4      '수신값이 없으면 MAIN으로
GOTO MAIN                         'MAIN 라벨로 분기 한다.
END
KEY1:                             '수신된 값이 1이면 처리
        ..................
GOTO MAIN                         '수신상태로 가기
KEY2:                             '수신된 값이 2이면 처리
        ..................
GOTO MAIN                         '수신상태로 가기
KEY3:                             '수신된 값이 3이면 처리
        ..................
GOTO MAIN                         '수신상태로 가기
KEY4:                             '수신된 값이 4이면 처리
        ..................
GOTO MAIN                         '수신상태로 가기
```

적외선 리모콘 1 번

새로 추가된 적외선 리모콘 1번은 기존의 리모콘이 단 1개의 로봇만을 제어할 수 있는 것에 비해, 제어할 수 있는 로봇의 ID 값을 총 4개까지 다르게 설정할 수 있어 여러 대의 로봇을 이용한 로봇 대회, 게임 등이 가능하다. 특히, 이족 보행 로봇 "로보노바(ROBONOVA-I)" 의 제어에 효과적이다.

이족보행 로봇아래 그림과 같은 모양이며, 각 버튼에 대한 코드 값은 다음과 같다.

MR-IR3 리모콘 모양

MR-IR3 리모콘 키 코드 표

키 번호	기능	ID 에 따른 키 값			
		ID-1	ID-2	ID-3	ID-4
K01	P1	25	57	89	121
K02	P2	19	51	83	115
K03	POWER	16	48	80	112
K04	1	1	33	65	97
K05	2	2	34	66	98
K06	3	3	35	67	99
K07	4	4	36	68	100
K08	5	5	37	69	101
K09	6	6	38	70	102
K10	7	7	39	71	103
K11	8	8	40	72	104
K12	9	9	41	73	105
K13	*	22	54	86	118
K14	0	10	42	74	106
K15	#	24	56	88	120
K16	▲	11	43	75	107
K17	◀	14	46	78	110
K18	■	26	58	90	122
K19	▶	13	45	77	109
K20	▼	12	44	76	108
K21	A	15	47	79	111
K22	△	21	53	85	117
K23	B	20	52	84	116
K24	◁	28	60	92	124
K25	□	29	61	93	125
K26	▷	30	62	94	126

K27	C	17	49	81	113
K28	▽	31	63	95	127
K29	D	27	59	91	123
K30	E	18	50	82	114
K31	F	32	64	96	128
K32	G	23	55	87	119

리모콘 수신 센서의 연결

- 적외선 수신 센서는 MR-C3000, MR-C3024 컨트롤러의 AD7 번 포트에 연결한다.

리모콘의 ID 설정방법

- P1 버튼과 변경하고자 하는 ID번호(1, 2, 3, 4 중 하나) 키를 약 2초간 동시에 누른다.

로보베이직의 프로그램 예)

```
'**** MR-IR3 프로그램 예제 ****
DIM IR_ID_CODE AS BYTE
CONST IR_ID = 1

'ID 체크하는 부분
IF IR_ID = 1 THEN
        IR_ID_CODE = 0
ELSEIF IR_ID = 2 THEN
        IR_ID_CODE = 32
ELSEIF IR_ID = 3 THEN
        IR_ID_CODE = 64
ELSEIF IR_ID = 4 THEN
        IR_ID_CODE = 96
ENDIF

MAIN:
```

```
        A = REMOCON(1)
        A = A - IR_ID_CODE

        ON A GOTO MAIN, KEY1, KEY2, KEY3, KEY4, ……
        GOTO MAIN
        END

KEY1:
        ……..
        GOTO MAIN

KEY2:
        ……..
        GOTO MAIN

KEY3:
        ……..
        GOTO MAIN
```

SONAR() 초음파()

초음파 포트에 연결된 초음파 센서로부터 계산된 거리를 읽어온다. 3000

명령어 구문

- 영문구문 : SONAR (*[초음파 포트]*)
- 한글구문 : 초음파 (*[초음파 포트]*)

명령어 설명

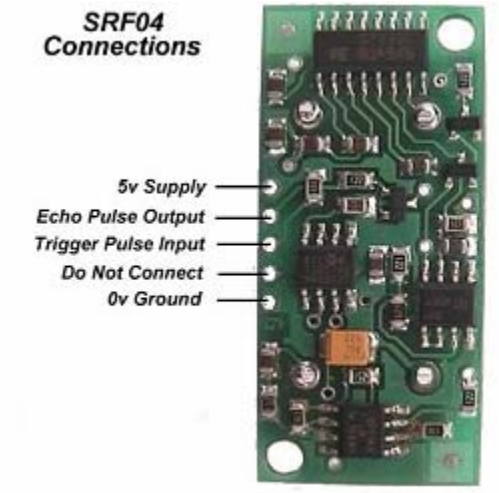
MR-C3000 계열 컨트롤러의 디지털 입출력 포트 0번부터 15번 포트까지를 초음파 포트 0번부터 7번까지로 사용하며, 아래 표와 같이 구성된다.

MR-C3000 계열 컨트롤러 디지털 입출력 포트 번호	초음파 포트
0번 포트	0번 초음파 포트 출력
1번 포트	0번 초음파 포트 입력
2번 포트	1번 초음파 포트 출력
3번 포트	1번 초음파 포트 입력
4번 포트	2번 초음파 포트 출력
5번 포트	2번 초음파 포트 입력
6번 포트	3번 초음파 포트 출력
7번 포트	3번 초음파 포트 입력
8번 포트	4번 초음파 포트 출력
9번 포트	4번 초음파 포트 입력
10번 포트	5번 초음파 포트 출력

11번 포트	5번 초음파 포트 입력
12번 포트	6번 초음파 포트 출력
13번 포트	6번 초음파 포트 입력
14번 포트	7번 초음파 포트 출력
15번 포트	7번 초음파 포트 입력

*SONAR의 값을 받는 변수는 반드시 정수형을 사용*한다. 즉, SONAR가 리턴하는 값은 0-3000까지 이며, 0이면 감지가 안 되는 경우, 그 외에는 mm 단위의 감지된 거리이다.

MR-C3000 계열 컨트롤러에서 사용 가능한 초음파센서는 ROBOT ELECTRONICS사의 SRF04모델이다.



초음파 센서의 연결은 MR-C3000(초음파출력) -> SRF04(Trigger Pulse Input), MR-C3000(초음파입력) <- SRF04(Echo Pulse Output) 이다.

명령어 예

```
DIM A AS INTEGER          'A라는 정수형 변수를 선언한다.
A = SONAR(3)              '초음파 포트 3번(디지털 입출력 포트 6,7번
                          '포트)를 이용하여 초음파 수신 값(거리)을 A라는
                          '변수에 저장한다.
```

RCIN() 조종기입력()

RC용 조종기와 수신기로부터 입력되는 펄스 값을 입력한다. 3000

명령어 구문

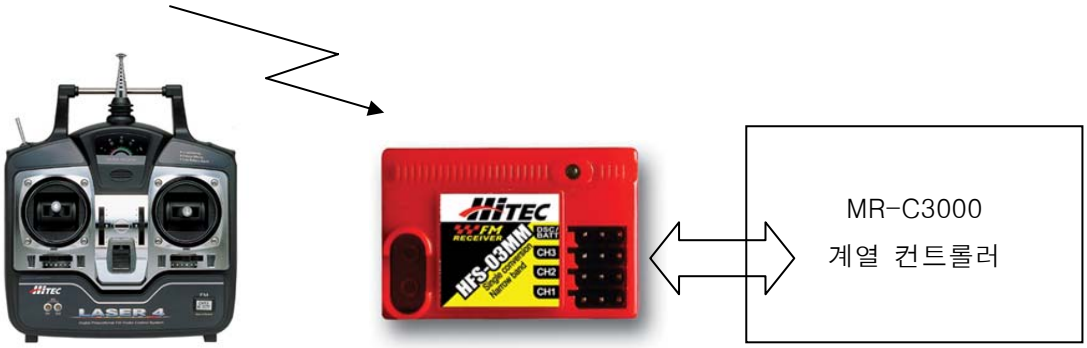
- 영문구문 : RCIN (*[RC수신포트]*)
- 한글구문 : 조종기입력 (*[RC수신포트]*)

명령어 설명

MR-C3000 계열 컨트롤러의 AD입력 포트에 RC용 조종기와 수신기를 부착하여 조종기의 수신 값을 읽어 들인다. 아래 표는 *[RC수신포트]*의 구성을 보여준다.

MR-C3000 계열 컨트롤러 AD 포트 번호 (디지털입출력 포트 번호)	RC수신 포트
0번 포트 (32번 포트)	0번 RC수신 포트
1번 포트 (33번 포트)	1번 RC수신 포트
2번 포트 (34번 포트)	2번 RC수신 포트
3번 포트 (35번 포트)	3번 RC수신 포트
4번 포트 (36번 포트)	4번 RC수신 포트
5번 포트 (37번 포트)	5번 RC수신 포트
6번 포트 (38번 포트)	6번 RC수신 포트
7번 포트 (39번 포트)	7번 RC수신 포트

MR-C3000 계열 컨트롤러에 연결하는 RC용 조종기는 AM방식 보다 FM방식을 사용하여야 전파혼선에 의한 오 동작을 방지할 수 있으며, 다양한 기능을 갖는 조종기를 사용하면 로봇 조종을 훨씬 다양한 방법으로 조종할 수 있다.



명령어 예

```
DIM A AS BYTE

A = RCIN(0)          'RC수신포트 0번에 연결된 수신기 신호를 A변수에
                     '저장
```

GYRODIR 자이로방향

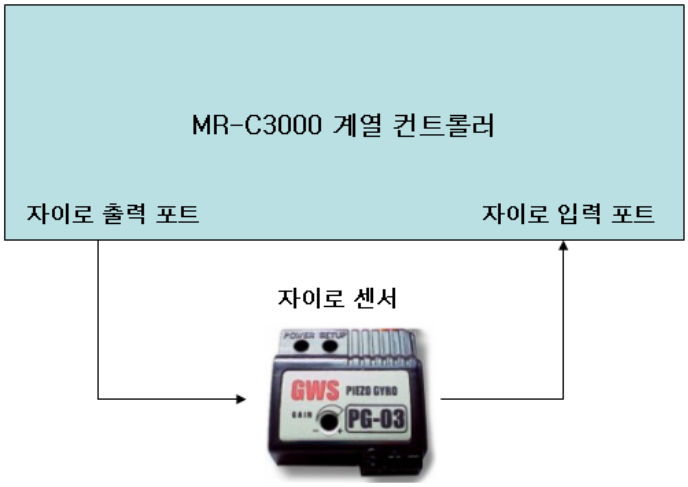
자이로 센서처리 서보모터의 동작 방향을 설정한다. 3000

명령어 구문

- 영문구문 : GYRODIR *[그룹지정], [모터n동작방향],……*
- 한글구문 : 자이로방향 *[그룹지정], [모터n동작방향],……*

명령어 설명

MR-C3000 계열 컨트롤러의 AD입력 포트에 자이로 센서를 연결하여 자이로 센서에 의한 처리 값을 포트에 연결된 각 서보모터에 반영할 때, 처리되는 동작 방향을 설정한다. 자이로 센서는 총 4개까지(자이로 포트 4개) 연결이 가능하며, 각 자이로 센서의 연결은 다음과 같다.



MR-C3000 계열 컨트롤러 AD 포트 번호 (디지털입출력 포트 번호)	자이로 포트
0번 포트 (32번 포트)	자이로 1번 채널 출력 포트
1번 포트 (33번 포트)	자이로 2번 채널 출력 포트
2번 포트 (34번 포트)	자이로 3번 채널 출력 포트
3번 포트 (35번 포트)	자이로 4번 채널 출력 포트
4번 포트 (36번 포트)	자이로 1번 채널 입력 포트
5번 포트 (37번 포트)	자이로 2번 채널 입력 포트
6번 포트 (38번 포트)	자이로 3번 채널 입력 포트
7번 포트 (39번 포트)	자이로 4번 채널 입력 포트

자이로 센서의 입력 값은 기준 값에서 회전하는 각도에 따라 좌우로 값이 변경되므로, 이에 대한 자이로 센서의 값을 서보모터에 반영할 때, GYRODIR 명령어를 사용하여 현재 서보모터 값에 자이로 입력 값을 가감할 지를 결정한다.

*[모터n동작방향]*은 “0” 또는 “1”을 사용하며, “1”은 자이로 입력값이 증가하면 현재 서보모터의 값을 증가시키고, “0”은 감소시킨다.

명령어 예

```
GYRODIR G6A, 1, 1, 0, 0, 1, 0
```

GYROSET 자이로설정

연결된 각 서보모터에 반영할 자이로 센서를 설정한다. 3000

명령어 구문

- 영문구문 : GYROSET *[그룹지정], [모터n자이로],……*
- 한글구문 : 자이로설정 *[그룹지정], [모터n자이로],……*

명령어 설명

MR-C3000 계열 컨트롤러에 연결할 수 있는 자이로는 총 4개이며, 컨트롤러에 연결된 서보모터 각각에 반영할 자이로 센서의 포트를 설정한다.

*[모터n자이로]*는 1부터 4에 해당되는 해당 서보모터에 반영될 자이로 센서의 포트 번호이다. 만약 0을 설정하게 되면 자이로 센서의 입력을 사용하지 않는다.

명령어 예

GYROSET G6B, 1, 1, 2, 2, 0, 0

'6번 서보모터는 자이로 센서 1번의 값을 입력받아 처리한다.
'7번 서보모터는 자이로 센서 1번의 값을 입력받아 처리한다.
'8번 서보모터는 자이로 센서 2번의 값을 입력받아 처리한다.
'9번 서보모터는 자이로 센서 2번의 값을 입력받아 처리한다.
'10번, 11번 서보모터는 자이로 센서를 사용하지 않는다.

GYROSENSE 자이로감도

연결된 각 서보모터에 반영할 자이로 센서를 설정한다. 3000

명령어 구문

- 영문구문 : GYROSENSE *[그룹지정], [모터n자이로감도],……*
- 한글구문 : 자이로감도 *[그룹지정], [모터n자이로감도],……*

명령어 설명

MR-C3000 계열 컨트롤러에 연결할 수 있는 자이로는 총 4개이며, 컨트롤러에 연결된 서보모터 각각에 반영할 자이로 센서의 감도를 설정한다.

*[모터n자이로감도]*는 0부터 255사이의 숫자, 상수를 사용하며, 0은 자이로 센서의 입력값을 전혀 반영하지 않으며 값이 커질수록 서보모터는 설정된 장로 센서의 입력값에 의한 빠른 반응을 보이게 된다.

명령어 예

GYROSENSE G6A, 100, 100, 255, 255, 50, 50

‘0번, 1번 서보모터는 자이로 감도를 100으로 설정한다.

‘2번, 3번 서보모터는 자이로 감도를 최대(255)로 설정한다.

‘4번, 5번 서보모터는 자이로 감도를 작게(50) 설정한다.

제 12 장

로보베이직

처리 명령어와

기타 명령어 설명

ON…GOTO 처리…가기

변수의 값에 따라 조건분기한다.

명령어 구문

- 영문구문 : ON *[변수]* GOTO *[행라벨], [행라벨], ...*
- 한글구문 : 처리 *[변수]* 가기 *[행라벨], [행라벨], ...*

명령어 설명

*[변수]*의 값에 따라 처리하고자 하는 프로그램이 달라질 경우에 사용되는 명령어가 ON...GOTO (처리...가기)이다. 값에 따른 분기는 IF(만약)문을 사용할 수 있으나 간단히 ON...GOTO (처리...가기)문으로 처리할 수 있으며, IF문에 비해 더 작은 크기의 목적코드를 생성해 낼 수 있다.

다음은 같은 기능을 수행하는 프로그램에 대해 IF문과 ON...GOTO문의 비교를 나타내었다.

```
A = BYTEIN(0)

IF A = 0 THEN
   GOTO 10
ELSEIF A = 1 THEN
   GOTO 20
ELSEIF A = 2 THEN
   GOTO 30
ELSE
   GOTO 40
ENDIF
```

```
A = BYTEIN(0)

ON A GOTO 10, 20, 30
IF A>2 THEN GOTO 40
```

GOTO(가기) 다음에 사용되는 *[행라벨]* 들은 *[변수]*가 0일 때부터 처리되며, 순차적으로 1씩 증가하는 변수 값과 대응된다.

*[변수]*에는 숫자, 상수, 변수를 사용할 수 있으며, *[행라벨]*의 개수는 최대 255까지 사용할 수 있다.

RND 난수

난수(무작위 수)를 발생한다.

명령어 구문

- 영문구문 : RND
- 한글구문 : 난수

명령어 설명

미니로봇컨트롤러를 이용한 로봇의 효과적인(랜덤한) 프로그램을 위하여 RND(난수) 명령어를 지원하며, 여기서 발생되는 난수는 매 실행 시마다 전혀 다른 수를 발생시키는 구조로 되어 있다.

난수는 0~255사이의 임의의 수를 발생시키는 함수이다.

명령어 예

```
DIM A AS BYTE

A = RND
BYTEOUT 0, A          '바이트포트 0번에 무작위 값이 출력된다.
```

REMARK 설명

설명문을 기입한다.

명령어 구문

- 영문구문 : REMARK *[설명문]*
- 한글구문 : 설명 *[설명문]*

명령어 설명

로보베이직을 이용하여 프로그램을 작성할 때, 프로그램의 작성에 대한 설명을 기입할 경우 간단히 기호 (') 를 사용하거나 REMARK(설명) 명령어를 사용한다.

*[설명문]*은 한 줄 범위 내에서 설명을 기입한다. 설명문은 프로그램의 내용에는 전혀 영향을 미치지 않는다.

명령어 예

```
REMARK 여기는 8개 LED를 모두 켠다.
BYTEOUT 0, 0
```

ACTION 행동

번호 값에 따라 정해진 템플릿(templet) 동작을 수행한다.

명령어 구문

- 영문구문 : ACTION *[번호]*
- 한글구문 : 행동 *[번호]*

명령어 설명

*행동[번호]*의 값에 따라 미리 정해져 있는 동작을 수행한다. 정해져 있는 동작의 개수는 총 32개이다.

명령어 예

ACTION 3 *'3번 행동 실행.*
ACTION 5 *'5번 행동 실행.*
ACTION 23 *'23번 행동 실행.*

주) 본 명령어는 MR-C3024와 로보노바 로봇 전용 명령어 입니다.

AUTO 자동

템플릿(templet) 프로그램으로 전환한다.

명령어 구문

- 영문구문 : GOTO AUTO
- 한글구문 : 가기 자동

명령어 설명

미리 정해져 있는 기본 퍼포먼스 영역(템플릿 영역)의 동작 프로그램을 실행시키기 위한 명령어이다.

명령어 예

```
GOTO AUTO                ' 템플릿 프로그램으로 이동.
```

주) 본 명령어는 MR-C3024와 로보노바 로봇 전용 명령어 입니다.

ACTION	key	동작 내용	변수	코드
0	power	ON : motor on->기본자세 OFF: 앉은자세-> motor off	A16	16
1	1	인사 -> 기본자세		1
2	2	팔 위로 올리고 -> 기본자세		2
3	3	앉기 -> 기본자세		3
4	4	앉고 -> 팔위로 -> 기본자세		4
5	5	한쪽다리 들기 동작 -> 기본자세		5
6	6	발벌리기 -> 팔 좌우벌리기 -> 좌우기울기 -> 기본자세		6
7	7	날개짓		7
8	8	양발 차기		8
9	9	물구나무서기		9
10	0	빨리 달리기		10
11	*	왼쪽 방향전환		22
12	#	오른쪽 방향전환		24
13	▲	전진		11
14	◀	좌로		14
15	■	앉기<->서기	A26	26
16	▶	우로		13
17	▼	후진		12
18	△	앞 덤블링		21
19	◁	좌 덤블링		28
20	□	정면 공격		29
21	▷	우 덤블링		30
22	▽	뒷 덤블링		31
23	A	왼 공격		15
24	B	우 공격		20
25	C	왼 앞 찌르기		17
26	D	오른 앞 찌르기		27
30	P1	일어나기 앞		25
31	P2	일어나기 뒤		19

	E,F,G	27,28,29 예약 18,32,23		18

제 13 장

로보베이직

의사 명령어 설명

‘$DEVICE ‘$컨트롤러

현재 작성하는 프로그램에 적용되는 컨트롤러를 설정한다.

명령어 구문

- 영문구문 : ‘$DEVICE *[컨트롤러종류]*
- 한글구문 : ‘$컨트롤러 *[컨트롤러종류]*

명령어 설명

로보베이직을 이용하여 프로그램을 작성할 때, 프로그램이 동작하는 컨트롤러를 프로그램에 직접 지정할 때 사용한다. 현재는 사용하는 *[컨트롤러종류]* 에 "MRC2000"과 "MRC3000"를 사용할 수 있다.

작성된 프로그램을 컴파일하게 되면 로보베이직 메뉴의 현재 "컨트롤러 종류"가 DEVICE명령에 사용된 컨트롤러로 자동 변경된다.

명령어 예

```
'$DEVICE  MRC2000    '현재 작성하는 프로그램은 MR-C2000 계열 컨트롤러를
                     ‘위한 프로그램이다.
```

'$LIMIT '$제한

현재 작성하는 프로그램에 사용되는 서보모터의 동작 범위를 제한한다. 3000

명령어 구문

- 영문구문 : '$LIMIT *[모터번호], [최소값], [최대값]*
- 한글구문 : '$제한 *[모터번호], [최소값], [최대값]*

명령어 설명

이 LIMIT(제한)명령어는 로보베이직의 실시간 모터제어 화면 중에 효과가 나타난다. 즉, 서보모터로 구성된 로봇을 실시간으로 제어하는 도중에 제한 범위 이상을 벗어나 로봇의 모터가 손상되는 것을 막아주기 위함이다. 그러므로, 프로그램을 컨트롤러에 다운로드하여 실행하는 것과는 관계가 없다.

*[모터번호]*에는 0번부터 31번까지 서보모터 번호를, *[최소값]*과 *[최대값]*에는 10에서 190사이의 모터 제한 범위 값을 사용한다. 초기에는 모든 모터가 10-190의 제한 범위를 갖는다.

명령어 예

```
'$LIMIT  0, 50, 100      '0번 서보 모터의 동작 각도 범위를 50에서
                         '100사이로 제한한다.
```