모션 마스터
(Motion Master V01)

-R13.338 버전 이상-

(2018-02-24 후)

V스크립트 기초문법

2015-09-29

Document History

Date	Descriptions	Version
18/02/28	[배포]	R01

[차례]

1. V스크립트와 모션마스터의 관계

모션마스터(Motion Master)는 CNC기술기반의 위칸 모션컨트롤러제품들을 운영하는 기본 소프트웨어입니다. CNC 기술은 컴퓨터기술을 이용하는 수치제어 모션기술 전반을 이야기 하지만 여기서는 G코드, M코드들로 구성된 NC코드체계에 의해 기계모션동작을 운영하는 기술로서 협의적 의미로 사용합니다.

모션마스터는 CNC기반이지만 공작기계뿐만 아니라 다양한 자동화머신 및 사용자기계에 응용될 수 있도록 아래와 같은 프로그램 방식들을 제공합니다.

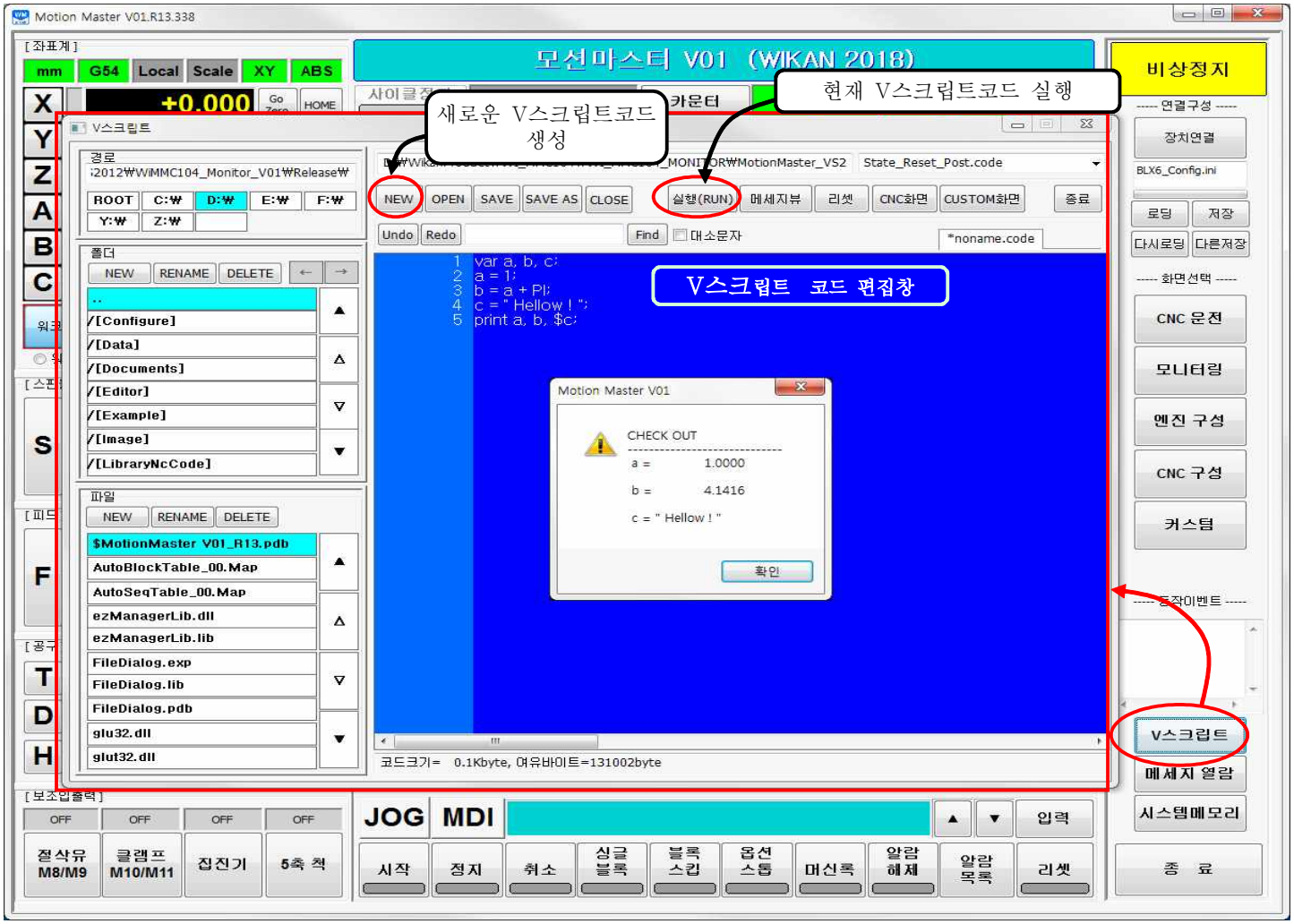
(1) 기본적인 NC코드 동작.
(2) 매크로 테이블에 의한 사용자 정의 NC코드 동작.
(3) NC코드의 제어확장문법(IF, WHILE, GOTO)에 의한 조건제어, 반복연산제어 기능.
(4) V스크립트에 의한 언어적 프로그램 기능.
(5) V스크립트에 의한 사용자화면 생성 및 실행.

이번 매뉴얼은 V스크립트 언어의 기초문법 이해를 목표로 합니다.
각 장별의 예제를 하나씩 실행해 가면서 기초문법과 해당 동작들을 이해해 봅니다.
(*모션 마스터의 화면이미지는 모션마스터의 버전에 따라 조금씩 다를 수 있습니다)

2. V스크립트 특징과 기초문법 #1

2.1. 무작정 따라해보기

(1) 모션마스터를 실행합니다. (모션보드가 있다면 장치연결 후 사용하시고, 없을 경우 모션보드동작과 관련된 동작에서 이상이 발생될 수도 있지만, 전체적인 문법학습은 가능합니다.)
: 우측의 탭화면 하단부에 [V스크립트] 라는 버튼을 클릭하면 아래와 같은 "V스크립트"실행윈도우가 생성됩니다.
-참고사진-



(2) V스크립트윈도우의 [NEW]버튼을 누르고 코드편집창에 명령코드를 아래와 같이 입력합니다..

```
var a, b, c;
a = 1;
b = a + PI;
c = " Hellow ! ";
print a, b, $c;
```

실행(RUN)

(3) [실행(RUN)] 버튼을 누르면 우측의
변수 확인창 이 나타납니다.

2.2. V스크립트 언어 특징.

V스크립트언어는 모션마스터 프로그램 환경 내에서 동작되는 단순한 문법체계를 갖는 라인-인터프리터 방식의 언어입니다. 인터프리터란 문자들을 하나씩 읽어 들임과 동시에 실행한다는 의미이며, 라인 인터프리터는 한 줄씩 한 줄씩 읽어서 실행한다는 의미입니다. 그러므로 하나의 실행문은 한줄 내에서 완전히 종료해야 합니다.

한 줄 내에 여러 실행문을 작성할 때 세미콜로 ';' 을 사용하여 아래와 같이 구분할 수 있습니다.
a = 1 ; b = a + PI ; c = "안녕하세요" ; print a, b, $c ;

그 외 특징들을 아래에 간략히 기술합니다.

(0) V스크립트 언어는 c언어의 문법과 유사.
(1) 소스코드 파일은 *.code 확장자를 갖는 아스키코드(혹은 멀티바이트코드) 문자파일.
(2) 모든 키워드는 대소문자 구분.
(3) 파일단위를 함수처럼 사용.
(4) 데이터형에 관계 없는 단일화 된 변수선언.
(5) C언어 형태의 주석문(설명문) 지원 //, /* ~ */
(6) 다양한 수학연산 문법과 400~500여개의 기본 내장 함수.
(7) PLC시퀀스 제어동작 지원을 위한 내장함수지원, X(입력), setY(출력), TON(타이머), M(비트메모리)....
(8) 모션보드 직접제어, 사용자 화면문법 및 사용자 3d드로잉 함수 등 지속적 함수지원
(9) 윈도우 표준 DLL 연결지원으로 고속 연산처리 및 다른 프로그램모듈과 연동가능.

2.3. V스크립트 실행 제한.

V스크립트 언어는 내부적인 인터프리터 실행성능에 의해 코드의 크기 및 사용자 변수 선언크기 등 몇 가지 제한을 갖고 있습니다. 큰 규모의 프로그램 코드를 구성할 경우 아래의 제한 사항을 고려해야 합니다.

(1) 프로그램 실행파일(소스파일) 크기 제한 : 128kbyte
(2) 1라인 문자 크기 제한 : 최대 2kbyte
(3) 제어문 { } 코드블록 크기 제한 : 최대 64kbyte
(4) 변수선언 제한 : 최대 1000개
(5) 배열변수 선언은 최대 2차원배열까지
(6) 내부변수 크기 제한 : 32kbyte
(7) 내부 문자열 선언 제한 : 32000개
(8) 내부 문자열 크기 제한 : 128kbyte
(9) 사용자 함수 정의 제한 : 최대 200 개
(10) 변수 및 함수명은 최대 30바이트 이내.
(11) 함수인자는 최대 15개 이내

V스크립트언어의 프로그램 조직화 방식은 파일단위입니다. 소스파일의 크기가 제한되어 있어 하나의 파일에 큰 규모의 프로그램을 작성할 수는 없습니다. V스크립트 언어는 소규모의 기능단위들을 파일로 나누어 프로그램 한 후 call()함수를 통해 필요한 기능들을 호출함으로써 전체 프로그램 동작을 구현합니다.

2.4. 변수 와 연산

- 변수는 사용자가 임의 이름으로 지정할 수 있는 기억공간입니다.
- 연산은 상수 혹은 변수를 대상으로 산술, 비교, 논리 등의 다양한 수학적인 처리를 말합니다.
- PI (원주율) 는 3.141592654...의 특정 값으로 정해져있고 이름이 내정되어 있는 변수로써 이를 상수형 변수라고 합니다.

2.4.1. V스크립트에서의 상수형 변수 목록

아래의 변수들은 상수형 변수로써 값을 변경할 수 없는 변수들입니다. 이들은 c언어의 매크로정의와 동일합니다.

순번	변수명	값	의미	C언어 대응 매크로 정의
1	M_E	2.71828182845904523536	네이피어수(자연대수) e	M_E
2	M_LOG2E	1.44269504088896340736	log2(e)	M_LOG2E
3	M_LOG10E	0.434294481903251827651	log10(e)	M_LOG10E
4	M_LN2	0.693147180559945309417	ln(2)	M_LN2
5	M_LN10	2.30258509299404568402	ln(10)	M_LN10
6	PI	3.14159265358979323846	원주율 (π)	M_PI
7	PI_2	1.57079632679489661923	(π/2)	M_PI_2
8	PI_4	0.785398163397448309616	(π/4)	M_PI_4
9	M_1_PI	0.318309886183790671538	(1/π)	M_1_PI
10	M_1_PI	0.636619772367581343076	(2/π)	M_2_PI
11	M_2_SQRTP	1.12837916709551257390	2/√π	M_2_SQRTPI
12	M_SQRT2	1.41421356237309504880	√2	M_SQRT2
13	M_SQRT1_2	0.707106781186547524401	1/√2	M_SQRT1_2
14	TRUE	1.0	참	TRUE
15	FALSE	0.0	거짓	FALSE
16	NULL	0	무효	NULL

2.4.2. 수치 표현 과 문자, 문자열 표현 방법, 이스케이프시퀀스

V스크립트언어는 단일 변수로써 정수(십진법/16진법/2진법), 실수, 문자, 문자열을 모두 사용합니다.
정수형 데이터는 최대 32비트 영역까지만 표기 가능합니다.

(1) 십진법 정수 표기 : 123, -123, +123
(음수의 표기 경우 빼기연산자와 구분이 필요함으로 가능하면 괄호()를 사용하여 (-123)으로 표기해줍니다.)

(2) 이진법 정수 표기 : 0b0001, 0b0000_1010, 0b01_01_01_01, 0B0011_0011
_표기를 이용하여 숫자간의 가독성을 편리하기 표기할 수 있습니다. 0b 혹은 0B 와 같이 첫문자가 숫자0으로 시작해서 대소문자 관계없이 다음 알파벳 b로 시작됩니다.

(3) 16진법 정수 표기 : 0x1234, 0x1234_5678, 0xFFCA_1234
0x혹은 0X의 대문자표기가 가능하며 a,b,c,d,e,f 는 대문자표기 A,B,C,D,E,F로 표기할 수 있습니다.

(4) 일반적인 실수 표기 : 12.34 -12.34 -0.123
(음수의 표기 경우 빼기연산자와 구분이 필요함으로 가능하면 괄호()를 사용하여 (-0.123)으로 표기해줍니다.)

(5) 과학적 실수 표기 : 12.34e3, 12.34e-2, -12.34E-5
e 혹은 E는 10의 지수승 표기법으로 지수승은 정수로만 표현되고 실수로 표현될 수 없습니다. 실수로 표현되더라도 정수로만 계산되어 집니다.
0.11e-1.2 => 0.11e-1 과 동일하게 취급됨으로 지수승도 실수로 표현하고 한다면 0.11*10^(-1.2) 의 연산표현법을 사용할 수 있습니다. (^는 누승연산기호)

(6) 문자 표기 : 'a', 'b', 'c', '+ ', '1', '2', ... '₩t', '₩v', '₩n', '₩b', '₩₩'
작은 따옴표를 둘러싸인 하나의 문자를 코드로 받아들입니다. ₩로 시작되는 문자표기를 이스케이프시퀀스라고 하며 이들은 일반적인 문자입력으로 표현될 수 없어 ₩로 시작한 후 다음 문자에 의해 코드를 결정하게 됩니다.
종류는 아래와 같습니다.

순번	이스케이프 시퀀스	종류	아스키 코드값(16진수)	
1	₩a	경고(벨)	0x07	
2	₩b	백스페이스	0x08	
3	₩f	폼피드	0x0C	
4	₩n	줄바꿈	0x0A	
5	₩r	캐리지 리턴	0x0D	
6	₩t	가로 탭	0x09	
7	₩v	세로 탭	0x0B	
8	₩'	작은 따옴표	0x27	
9	₩"	큰 따옴표	0x22	
10	₩₩	백슬래시	0x5C	
11	₩?	물음표	0x3F	
12	₩;	세미콜론	0x3b	
13	₩xhh	16진표기 아스키코드값	예> ₩x61 은 문자'a' 와 동일	

(7) 문자열 표기 : "Hellow !", "Hellow! ₩t Mr.Kim",
V스크립트에서 지원하는 기본적인 문자 와 문자열은 기본적으로 멀티바이트 코드입니다.

부득이 유니코드 문자열을 필요하다면 _L()함수를 사용할 수 있습니다. 유니코드 문자열을 멀티바이트코드 문자열로 바꾸려면 _M()함수를 사용할 수 있습니다.
예: 유니코드변환> var unicode_str; unicode_str = _L("Hellow");
예: 멀티바이트코드변환> var multibyte_str; multibyte_str = _M(_L("Hellow"));

2.4.3. 변수 선언문

V스크립트의 변수는 var 라는 단일 명령어를 사용하여 선언합니다. (대문자 VAR도 가능)
변수의 이름은 첫 글자가 숫자가 아닌 문자로써 시작되며, 최대 30자 이내입니다.
한글변수도 사용할 수 있지만, V스크립트내에서 사용되는 명령어나 함수이름과 중첩되지 않아야 합니다.
V스크립트의 변수는 8바이트 단위의 배정도실수형(C언어의 double에 해당)입니다.

(1) 일반적인 단일변수 선언

```
var a, b, c;  // 각각 a, b, c라는 이름의 변수 선언문.

var aa(1); // aa라는 변수를 선언하고 초기값을 1로 설정.
```

단일 변수는 선언과 동시에 0의 값으로 초기화 됩니다. 다른 값으로 초기화 할 경우 괄호() 안에 상수 수치값을 기입합니다. 괄호() 내부에는 수식(표현식) 등으로 표현될 수 없고, 단지 1, 12.34 의 등의 10진수나 0x34와 같은 16진수, 혹은 0b1101와 같은 2진수 상수값으로 기입해야 합니다.

(2) 1차원 배열 변수

```
var a[3], b[6]; // 각각 배열크기가 3, 6인 변수 a, b를 선언
var c[9](123);  // 배열 크기가 9인 변수 c를 선언하고 123으로 전체를 초기화

a[0] = 1;       // 각 배열요소 마다 값을 지정.
a[1] = 2;
a[3] = 3;

a = ( 2, 3, 4 );                // 한 번에 각 배열요소들의 값을 지정.
b = ( 1, a, 5, 6 );             // 다른 변수의 내용을 포함하여 각 배열요소들의 값을 지정.
c = ( a, b );
```

a = (2, 3, 4)와 같은 =대입문의 (,,,)방식은 아래와 같이 수식이 포함될 수 있습니다.
a = (2 * sin(PI), 3-4, 4);

※ 배열변수로 선언 할 경우 전체 배열요소의 크기가 1보다는 커야 합니다.
즉 var a[1]; 의 선언은 var a의 선언과 동일함으로 정확히 선언하기 위해서 적어도 배열요소의 수가 1보다는 큰 var a[2]; 로 선언해야만 a라는 변수를 배열변수로 취급하게 됩니다.

(3) 2차원 배열 변수
V스크립트는 최대 2차원 변수까지만 지원합니다.

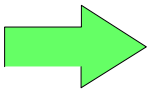
```
var a[3], b[6];          // 각각 배열크기가 3, 6인 변수 a, b를 선언
var c[3][3](123);        // 배열 크기가 3x3인 2차원 배열변수 c를 선언하고 123으로 전체를 초기화

a = ( 1, 2, 3 );                 // 한 번에 배열요소들의 값 지정.
b = ( a, 4, 5, 6 );
c = ( a, b );
c[0] = a;       // c[0]행의 첫 배열요소부터  a배열의 값 1,2,3 값을 차례대로 대입.
c[1] = b;       // c[1]행의 첫 배열요소부터  a배열의 값을 순차적으로 대입,
                // 단, c[1]의 행의 크기와 관계없이 c변수와 b변수의 크기만큼  대입.
```

array[r][c] 의 형태로 2차원 배열변수를 선언하며, r을 행(row), c를 열(column) 이라고 합니다.
메모리는 열 우선으로 순차적으로 배열됩니다.
즉 대입문으로 arrary = (1,2,3); 으로 지정되면 array[0][0]=1, array[0][1]=2, array[0][2]=3의 순서로 지정됩니다.

2.4.4. 표현식 (Expression)

V스크립트는 기본적으로 한 줄씩 읽어서 처리하는 라인인터프리터입니다. 이때 하나의 줄에 완전한 하나의 표현식(Expression)이 기술되어야 하며 아래와 같이 두 줄 이상에 걸쳐 표현될 수 없습니다.

```
// 오류 표현식
var a;
a =     a
        +
        10;
```



```
// 정상 표현식
var a;
a = a + 10;
```

표현식은 연산자(=, +, -, *, /...)와 연산대상자(상수, 변수)로 나열된 문입니다. 대부분 연산을 목적으로 하는 일련의 수식과 비슷하게 사용하는 경우가 많으므로 수식 혹은 연산식 이라고도 표현하겠습니다.

(1) 표현식의 예

```
var x, y, z, avr;              // (변수)선언문
x = 10;  y = 20; z = 30;       // 한 줄에 3개의 표현식 나열
avr = (x + y + z ) / 3;        // x,y,z 평균을 구하는 표현식
print avr;                     // 프린트문 (평균값 확인)
```

(2) 사용 가능한 내부 연산자와 우선순위

우선 순위 값은 작을수록 높은 우선순위를 나타냅니다. 연산 우선순위가 모호할 때는 괄호()를 사용하여 확실하게 우선순위를 정할 수 있습니다.

예) y = a + b * c ^ e | f ; => 우선순위에 따른 동일한 괄호표기 => y = ((a + (b * (c^e))) | f) ;

종류	기호	우선순위	설명
대입연산자	=	10 (가장낮음)	A = B // B값을 A에 복사
산술연산자	+	4	덧셈 A = B + C
	-	4	뺄셈 A = B - C
	/	3	곱셈 A = B * C
	*	3	나눗셈 A = B / C
	%	3	나머지계산 (나눈 후 정수 몫을 제외한 나머지값) A = B % C
승수연산	^	2	A = B ^ C ; // 2^3 => 8
괄호	( )	0 (가장높음)	연사자의 우선순위에 관계없이 ( )안이 먼저 실행됩니다.

증감연산자	++	1	A++; // A = A + 1;
			주의> ++A 순서로 사용할 수 없음. 배열변수에 사용할 수 없음.

	--	1	A--; // A = A - 1;
			주의> --A 순서로 사용할 수 없음. 배열변수에 사용할 수 없음.

대입+산술연산자	+=	7	A += B; // A = A + B; 주의> 배열변수에 사용할 수 없음.
	-=	7	A -= B; // A = A - B; 주의> 배열변수에 사용할 수 없음.
	*=	7	A *= B; // A = A * B; 주의> 배열변수에 사용할 수 없음.
	/=	7	A /= B; // A = A / B; 주의> 배열변수에 사용할 수 없음.

	^=	7	A ^= B; // A = A ^ B

종류	기호	우선순위	설명
비트 연산자	|	5	비트 OR 연산 (16비트정수값으로 치환하여 비트연산을 수행합니다.)
	&	5	비트 AND 연산
	~	2	비트 NOT(반전)연산
	$	5	비트 XOR 연산
			(OR, AND, NOT, XOR 문자열 명령도 지원)
	<<	5	왼쪽 시프트
	>>	5	오른쪽 시프트
대입+비트연산자	|=	7	A|=B ; // A = A | B; 주의> 배열변수에 사용할 수 없음.
	&=	7	A&=B ; // A = A & B; 주의> 배열변수에 사용할 수 없음.
	$=	7	A$=B; // A = A $ B; 주의> 배열변수에 사용할 수 없음.
	<<=	7	A<<=B; // A = A << B; 주의> 배열변수에 사용할 수 없음.
	>>=	7	A>>=B; // A = A >> B; 주의> 배열변수에 사용할 수 없음.
논리 연산자	||	7	논리 OR 연산 (논리연산대상자가 0.0의 값이면 거짓, 아니면 참을 기준합니다.) (결과가 참이면 1.0을 거짓이면 0.0을 리턴합니다.)
	&&	7	논리 AND연산
	!	2	논리 NOT연산
	$$	7	논리 XOR연산
비교 연산자	==	6	A == B (A와 B값이 동일할 때 참(1.0), 아닐때 거짓(0.0)을 리턴)
	!=	6	A != B (A와 B값이 서로 다를 때 참, 같을 때 거짓을 리턴)
	>	6	A > B (A가 B보다 클 때 참, 작거나 같을 때 거짓을 리턴)
	>=	6	A >= B (A가 B보다 크거나 같을 때 참, 작을 때 거짓을 리턴)
	<	6	A < B (A가 B보다 작을 때 참, 크거나 같을 때 거짓을 리턴)
	<=	6	A <= B (A가 B보다 작거나 같을 때 참, 작을 때 거짓을 리턴)

(3) 기초연산에 유용한 내장함수들 #1

모든 함수의 우선순위는 1입니다. 즉 괄호()연산자 다음으로 높은 우선순위를 갖고 있습니다.

함수형	우선순위	설명
max(a,b)	1	c = max(a,b); // a,b중 큰값을 c에 복사
min(a,b)	1	c = min(a,b); // a,b중 작은값을 c에 복사
maxabs(a,b)	1	c = maxabs(a,b); // a,b중 절대값이 큰 값을 c에 복사
minabs(a,b)	1	c = maxabs(a,b); // a,b중 절대값이 작은 값을 c에 복사
absmax(a,b)	1	c = absmax(a,b); // a,b중 절대값이 큰 값의 절대값을 c에 복사
absmin(a,b)	1	c = absmin(a,b); // a,b중 절대값이 작은 값의 절대값을 c에 복사
minmax(a,b,c)	1	d = minmax(a,b,c); // b값이 a와 c범위(a<=b<=c)내의 가장 가까운 값으로 반환
range(a,b,c)	1	d = range(a,b,c); // b값이 a와 c범위(a<=b<=c)내 일경우 1, 아니면 0을 반환
tolerance(a,b,c)	1	d = tolerance(a,b,c); // a값이 >=(b-c) 이고 <=(b+c) 일때 1, 아니면 0을 반환
abs(a)	1	a의 절대값

(4) 기초 수학함수들

종류	함수명	우선순위	설명
삼각함수	sin(r)	1	모든 소문자 삼각함수는 각도단위를 라디안값으로 취급하고,
	cos(r)	1	대문자 삼각함수는 각도단위를 도 단위로 취급합니다.
	tan(r)	1	
	asin(r)	1	
	acos(r)	1	
	atan(r)	1	-π/2 ~ +π/2
	atan2(a,b)	1	-π ~ +π
	atan3(a,b)	1	0 ~ 2π
	rad(d), deg(r)	1	라디안값과 degree값의 변환
로그, 지수	ln(n)	1	대문자함수 LN, LOG, EXP도 사용가능합니다.
	log(n)	1	
	exp(n)	1	e^n
	pow(a,b)	1	a^b와 동일
	sqrt(a)	1	
	round(a)	1	소숫점이하를 반올림합니다.
	ceil(a)	1	소숫점이하를 올림합니다.
	floor(a)	1	소수점이하를 버립니다.
	mod(a)	1	소수부분을 얻습니다.
	BCD(a)	1	a값을 4비트 bcd값으로 변환합니다.
	BIN(a)	1	bcd코드 a값을 이진값으로 변환합니다.

2.4.5. 정적변수(svar)

svar(혹은 SVAR) 변수 선언문은 var 변수선언문과 형태는 동일하지만 정적변수(static:스태틱변수)로 선언하게 됩니다. svar 변수는 Background.code 혹은 Background_UI.code 와 같이 독립된 하나의 인터프리터가 동일한 파일을 계속해서 해석하는 경우에 사용될 수 있습니다. 정적변수는 상태천이에 의한 프로그램방식에서 상태변수로 사용하면 편리합니다.

-Backgroun.code- 파일내에서

```
svar state(0);          // 정적변수 선언 및 초기화 (최초 단 한번만 초기화됨)
if( state == 0 )
{
        ...;            // 상태 0처리
        state = 10;     // 상태 10으로 전환
}
else if( state == 10 )
{
        ...;            // 상태 10처리
        state = 20;     // 상태 20으로 전환
}
...
```

정적변수는 동일영역의 중복선언에서 오류를 발생하지 않습니다.

2.4.6. 변수선언문과 표현식 (Expression) 예

하나의 변수선언문 혹은 표현식은 한 줄 안에 완전히 표현되어야 하며 최대2000자까지 가능합니다.

```
var a, b, c;                          // 변수 선언
a = 2/3*PI ; b = 2;                   // 한 줄에 2개의 표현식 문
c = sin(b*a) / cos(a) + atan2(a, b);  // 한 줄에 1개의 표현식 문
print a, b, c;                        // 변수값 확인문
```

※ print 문> print 는 함수가 아닌 명령어로써 연이어 오는 변수들의 내용을 확인하는 창을 만듭니다.

2.4.7. 시스템 메모리 접근

v스크립트 언어가 모션마스터에 직접적인 영향을 주는 방식은 크게 두 가지로써 (1)하나는 모션마스터동작과 관련된 함수를 사용하는 방식이고 (2) 둘째는 모션마스터가 제공하는 시스템메모리를 액세스하는 방식입니다.

첫번째 방식의 예는 WINC_START() 함수처럼 미리 정의된 기능을 실행하는 전용함수를 호출하는 것입니다. 모션마스터 동작과 관련된 전용의 함수들은 별도의 레퍼런스 가이드를 참조하시면 됩니다.

다음으로 모션마스터가 제공하는 시스템메모리를 접근하여 리드/라이트 함으로써 모션마스터 동작을 제어하는 것입니다.

시스템 메모리를 액세스하는 간단한 방식은 #메모리번지 형식을 사용하는 것입니다. 아래의 예를 참조합니다.

```
var nc_pos[6];
nc_pos[0] = #30000; // 시스템 30000 번지는 워크좌표 X
nc_pos[1] = #30001; // 시스템 30001 번지는 워크좌표 Y
nc_pos[2] = #30002; // 시스템 30002 번지는 워크좌표 Z

#30000 = nc_pos[0]; // 와 같이 반대로 사용할 수도 있지만 시스템메모리에 따라서 해당메모리가 읽기전용일 수도 있습니다.
```

상기의 방식은 nc_pos[0] = ... 의 표현식이 실행되는 시점에서만 nc_pos라는 변수의 내용이 갱신됩니다.
반면에 아래의 방식은 특정변수가 항상 시스템메모리와 연결됩니다.

```
var nc_pos[6]:#30000;
```

변수선언과 동시에 해당변수를 시스템메모리와 연결하는 방식입니다. 이 방식은 시스템메모리를 숫자가 아닌 이름(변수명)으로 액세스하는 방식입니다. 이러한 방식의 변수선언은 변수를 위한 별도의 메모리를 생성하지 않고 시스템메모리를 그대로 사용합니다. 단지 변수의 메모리 위치를 시스템메모리위치로 옮기는 역할만 합니다.

2.5. 제어문

제어문은 분기 제어문과 반복 제어문으로 나누어 볼 수 있습니다.

분기제어문은 코드 실행흐름을 변화시키는 문을 말합니다. 실행을 이동시키는 것을 분기한다 혹은 제어이행한다. 라고 말합니다.

조건분기는 조건의 표현식(수식)을 참조하여 참일 때와 거짓 일 때 서로 다르게 분기하는 것을 말하며 대부분 if()/else if()/else 등의 구문에 의해서 구현됩니다.
무조건 분기는 goto문과 같이 조건없이 바로 분기하는 문을 말하며 v스크립트에서는 지원하지 않습니다.
다만 반복제어문내에서 break, continue와 같은 제어이행 명령은 지원합니다.
(break문: 반복문에서 탈출, continue문: 다음번 반복실행으로 이행)

반복제어문은 특정문장들을 원하는 만큼 반복 실행하는 문을 말합니다. 반복제어문 역시 조건반복과 무조건 반복이 있습니다. 무조건 반복은 조건반복의 조건상태가 항상 참이면 구현될 수 있으므로 큰 의미를 갖지 않습니다.
조건반복은 조건의 표현식이 참인 동안만 특정 구획의 문들을 반복 실행합니다. v스크립트의 while()문 과 loop()문에 의해 조건반복동작을 실행합니다.

2.5.1. 조건분기문 if() else if() else

괄호()안의 조건에 따라 실행을 이행하게 됩니다. 아래는 if문의 기본적인 구조를 나타냅니다.

```
if( 조건1 표현식 )
{
        조건1 참 실행문들;
}
else if( 조건2 표현식 )
{
        조건2 참 실행문들;
}
else if( 조건3 표현식 )
{
        조건3 참 실행문들;
}
...
...  원하는 만큼 else if(){ }문 반복
...
else
{
        모든 조건들이 만족되지 않을 때 실행할 문들;
}
```

필요에 따라 if(){...}문 단독 또는 if(){...} else if(){...} 문 혹은 if(){...} else {...} 문이 사용될 수 있습니다
이때 반드시 조건에 대한 실행문은 코드블록(중괄호) { } 에 둘러싸여 있어야 합니다.

2.5.2. whlie() 문

괄호()안의 조건이 참인 동안 코드블록{ }내의 문장들을 실행합니다.

```
while( 조건 표현식 )
{
        반복 실행문들;
}
```

2.5.3. break, continue문

반복제어문내에서 break문을 만나면 해당 반복제어문에서 가장 가까운 반복제어문을 종료합니다.

```
while( 조건1 )
{
        while( 조건2 )
        {
                break;
        }
        반복 실행문들;
}
```

가장 가까운 반복제어문에서 빠져나옴.

반복제어문내에서 continue문을 만나면 해당 반복제어문의 조건처리부터 다시 평가하여 반복을 시작합니다.

```
while( 조건1 )
{
        while( 조건2 )
        {
                continue;
        }

        반복 실행문들;
}
```

조건문 다시 평가 후 반복 시작

2.5.4. loop() 문

괄호()안의 표현식의 값을 정수화 하여 반복실행할 횟수(카운터)로 사용합니다.

```
loop( 반복실행할 횟수에 대한 표현식 )
{
        반복 실행문들;
}
```

예>

```
var cnt;
cnt = 100;
loop( cnt )
{
   cnt++;
}
```

loop문은 초기에 단 한번 괄호()의 안의 수식을 평가해서 반복할 횟수를 기억합니다. 그러므로 괄호()내의 수식에 영향을 줄 수 있는 변수들의 값이 변하더라도 반복실행에 영향을 주지 않게 됩니다.
단, continue문을 만나면 수식은 재평가되고 반복실행은 처음부터 다시 시작됩니다.

2.5.5. loop() 의 특징과 유의점!

loop()문은 괄호내의 표현식이 단 한번만 해석되기 때문에 whlie()문에 비해서 빠르게 실행되는 장점이 있습니다.
특히 특정횟수만큼만 반복해서 사용할 경우 별도의 변수 선언 등을 필요로 하지 않으므로 편리하게 사용될 수 있습니다.
loop()문의 코드블록{ }내에서도 break문과 continue문이 사용될 수 있습니다. 동작은 while() 반복문에서의 동작과 동일하지만 continue문을 만나면 반복문의 실행카운터가 0으로 초기화 되고, loop괄호()내의 수식이 재해석되어 새롭게 실행됩니다. 그러므로 아래와 같은 코드를 작성하면 무한 루프에 빠지게 됩니다.

```
loop(100)
{
        continue;
}
```

2.5.6. 무한루프연산을 방지하는 워치독타임 wdt() 함수

v스크립트는 반복제어문의 실행시간을 감시하여 코드실행을 중단하는 기능을 갖고 있습니다. wdt()함수로써 실행시간 감시기능을 활성화 할 수 있습니다.

wdt()함수의 괄호()내의 인자는 시간(초단위)값으로 해당 시간이 경과되면 wdt()아래의 반복제어문들은 설정된 시간 내에서만 동작하게 됩니다. 아래의 예제를 확인해 봅니다. wdt() 괄호내의 값이 0이면 감시기능을 사용하지 않음을 뜻합니다. 음수이면 실행시간이 경과될 경우 스크립트 실행알람과 함께 종료합니다.

```
var cnt;
wdt(0.1);
loop(100000)
{
        cnt++;
}
print cnt;
```

실행

2.6. 파일모듈 및 함수

V스크립트의 프로그램 모듈화 방식은 코드파일들의 소스 결합(include)과 호출(call)입니다.
모션마스터프로그램을 커스터마이징하기 위해서는 자신만의 커스터마이징 폴더를 ScriptCode폴더 내부에 생성하고 그곳에 다양한 기능을 위한 코드들을 파일단위로 만들어 필요할 때마다 해당 파일을 호출하는 방식으로 구조화 합니다.

모션마스터프로그램의 커스텀마이징을 위한 주요한(main) 실행함수는 아래 두 가지 파일입니다.

(1) 하나는 Background.code 파일이고,
(2) 나머지는 Show_Custom_UI.code 파일입니다.

Background.code 파일은 cnc운전과 병행하여 독립적으로 순환 처리되는 시퀀스 프로그램을 작성하는 파일이며,
Show_Custom_UI.code 파일은 사용자가 자신의 화면을 새롭게 만들고자 할 때 실행되는 파일입니다.
Show_Custom_UI.code 파일에는 기본적인 코드들이 이미 작성되어 있으며 이들 코드의 내용을 부분적으로 수정하여 자신이 만든 화면코드파일을 호출call()하는 방식으로 사용됩니다.

Background_UI.code 는 Background.code와 달리 조작화면(UI)의 메인동작(Show_Custom_UI.code)과 병행하여 UI와 관련된 특정 동작들을 주기적으로 실행할 목적으로 호출(대체로 0.1초마다)실행되는 파일입니다.

Background_Setup.code 파일과 Background_UI_Setup.code 파일은 리셋초기에 한번만 실행되어 각각 Background.code 실행과 Background_UI.code 실행의 셋업과정을 도와주는 실행파일입니다.

그외 다양한 v스크립트 실행파일(*.code)들이 모션마스터의 실행 상태에따라 자동으로 호출됩니다. 이러한 실행파일들을 편집하여 자신만의 특정동작을 구현할 수 있습니다.

상기와 같이 v스크립트 언어의 구조화 방식에서 자주 사용되는 함수가 include()함수와 call()함수 입니다.
예를 들면, 모션마스터의 [F7]버튼을 누르면 티칭위저드창(Teaching Wizard Window)이 생성됩니다.
(티칭위저드창 역시 v스크립트언어로 구성되어 있습니다)
[F7]버튼을 누르면 모션마스터는 ScriptCode폴더내에서 UserKey_F1.code와 UserKey_F1_UI.code 파일을 찾아 실행합니다. UserKey_F1_UI.code파일은 아래와 같이 call()함수를 사용하여 다른 파일(실제 티칭기능을 실행하는 메인코드파일)을 호출하여 실행하는 코드가 작성되어 있습니다.

UserKey_F1_UI.code 파일

```
call ("₩₩WizardCode₩₩Wizard_Teaching₩₩Teaching_UI.code") ;
return(0);
```

2.6.1. include()함수와 call()함수에 의한 파일모듈관리

include()함수 및 call()함수의 괄호()에는 경로를 포함한 파일명을 전달합니다. 파일명은 문자열로써 큰따옴표“ ”로 둘러싸여야 합니다. 그리고 폴더와 폴더의 경계구분의 역슬래시(₩) 사용시 이스케이프시퀀스에 의해 두개의 역슬래시(₩₩)문자를 사용해야 합니다.

전체 경로명 include()함수 예>
include("C:₩₩WIKAN₩₩MotionMaster₩₩ScriptCode₩₩CommonCode₩₩RGB.code");

상대경로명 call()함수 예>
call("₩₩UserKey_F1_UI.code");

include()/call()함수는 아래의 경로들을 기본적으로 검색영역에 포함하고 있어 아래의 기본경로명은 생략할 수 있고, 다음 상위 경로부터 표기할 수 있습니다. 이를 상대경로명 표기라 합니다.
- 모션마스터 설치폴더 > 예) C:₩₩WIKAN₩₩MotionMaster
- 모션마스터 설치폴더의 ScriptCode폴더 > C:₩₩WIKAN₩₩MotionMaster₩₩ScriptCode

include("codefile.h")함수는 codefile.h의 내용을 그대로 복사해서 포함시키게 됩니다.
include()함수로 포함된 내용은 전체 소스코드의 kbyte크기에는 포함되지 않지만, 인클루드파일내에서 포함된 변수선언 등은 그대로 반영됩니다. 그러므로 인클루드파일의 내용은 호출한 소스와 동등한 레벨에서 해석되어 집니다.
c언어에서의 #include 문맥과 같이 공통의 헤더파일(*.h) 내용을 결합하는데 사용합니다.
include 할 파일내에서는 return()함수를 사용하지 말아야 합니다. return()함수를 사용하면 include()를 호출한 소스코드도 함께 종료됩니다.

반면에 call("codefile")함수는 단독의 인터프리터를 생성해서 독립적으로 실행하게 됩니다.
그러므로 call()문에서 호출하는 소스와 호출되는 소스코드(codefile)는 상호 독립적이며, 변수선언 등이 서로 영향받지 않습니다. 또한 호출파일에서 return(x); 로 종료되면 x값을 반환하게 됩니다.
그리고 호출되는 코드파일에 인자를 전달하고 싶다면 아래의 2가지 방식을 사용할 수 있습니다.

(1) 시스템 메모리를 사용 : 0번지에서 19999번지는 사용자에 의해서 임의적으로 사용될 수 있는 시스템메모리 공간입니다. 이 공간을 적절하게 사용함으로써 호출되는 소스코드에 인자를 전달할 수 있습니다.

(2) 직접적으로 인자를 전달합니다. 이때 최대 전달될 수 있는 인자수는 14개까지 입니다.
인자 전달 예>

호출하는 소스코드

```
var a(1), b(2), c;
c = call ("Called.code", a, b) ; // a, b인자를 전달.
print c;
```

호출되는 소스코드 ("Called.code")

```
arg para1, para2; // 전달받을 인자변수를 선언.
return( (para1 + para2) );
```

2.6.2. function 문에 의한 사용자 함수 정의

소스코드 내에서 사용자 함수를 정의할 수 있습니다. 함수는 작성된 파일내에서만 유효합니다.

```
function 사용자함수명( 인자1, 인자2, ... )
{
        arg 인자변수명1, 인자변수명2;

        .... (코드본체);

        return (결과값);
}
```

인자1, 인자2 와 인자변수명1, 인자변수명2 의 이름은 동일할 필요 없습니다. 실제 인자1, 인자2는 사용자함수를 등록할 때 단순히 몇개의 인자를 사용하는지에 대해서 알려줄 뿐입니다. 실제 함수가 실행 될 때는 실시간적으로 전달받은 인자들을 arg문에 의해 인자변수들로 각각 할당하게 됩니다. 이때 전달되는 인자와 동일한 형태(배열관계)의 변수로 선언해야 합니다. ... 으로 인자를 선언하면 가변인자로써 최대 15개까지 가능합니다.

예> 배열인자를 받아서 평균을 구함.

```
function avr( buf, max )
{
        arg buf[10], max;
        var cnt, sum, average;

        loop(max)
        {
                sum += buf[cnt];
                cnt++;
        }
        average = sum / cnt;
        return( average );
}

var data[10]:#00000, average;

average = avr(data, 10);

print average;
```

함수 내부에 또 다른 함수를 정의할 수 있습니다. 함수 정의문은 어디든 가능하지만 반복 제어문 내에서 선언하게 되면 반복해서 정의되므로 오류를 발생합니다.

※ 사용자 정의 함수(function)와 코드파일호출(call)의 특징.

(1) 사용자 정의함수를 포함하는 소스코드에 정의한 함수 밖의 다른 변수들을 액세스 할 수 없습니다.
만약 함수내에서도 외부변수를 액세스해야하는 경우는 시스템메모리의 사용자영역(#00000~#19999)을 사용해야 합니다.

(2) 사용자 정의함수의 실행속도가 코드파일호출보다 빠르게 실행됩니다.

2.7. 문자열 처리

V스크립트는 별도의 문자열이나 문자를 저장하는 변수형이 없으며 아래와 같이 var 단일 변수선언으로 문자와 문자열을 처리합니다.

2.7.1. 간단한 문자열 처리 예제

```
var str1, str2, str3;
str1 = "WIKAN";
str2 = "-Motion Controller"
str3 = strcat( str1, str2 );
print $str3;
```

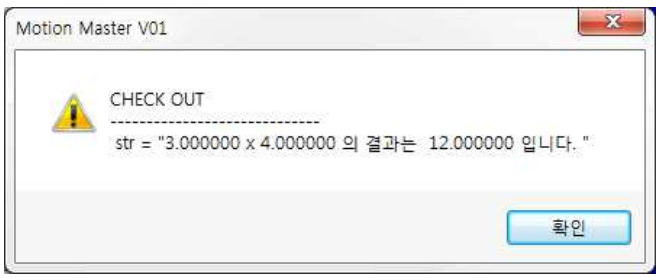
실행

str1 = “WIKAN"; // 의 실행은 내부의 문자열테이블에 "WIKAN"이라는 문자열을 등록하고 문자열테이블 인덱스값을 돌려줍니다.
print $str3; 문장을 print str3; 으로 바꾸면 str3값에 저장된 숫자값이 표시됩니다. 이 숫자값이 내부 문자열테이블에 등록된 인덱스값입니다. 내부 문자열에 등록될 수 있는 최대크기는 128kbyte까지이며 문자열의 갯수는 최대 32000개까지 입니다. 동일문자는 새롭게 등록되지 않으며 이미 등록된 문자열테이블 인덱스값을 돌려 줍니다.

다양한 문자열처리 함수들이 있으며 추후 V스크립트 버전에 따라 추가될 수 도 있습니다.
sprintf()함수를 사용한 예>

```
var a, b, str;
a = 3;
b = 4;
str = "결과는 “
str = sprintf("%f x %f 의 %s %f 입니다. ", a, b, str, a*b);
print $str;
```

-실행결과-



sprintf(“format string", var...); // 함수는 c언어의 포맷문자열과 동일합니다. format string에 대응하는 변수들을 전달하여 전제 문자열을 생성합니다. 이때 변수의 수는 최대 14개까지만 가능합니다.

2.7.2. 문자열 처리 함수들

종류	기호	설명
	_L(s1)	멀티바이트문자열s1을 유니코드로 변환한다.
	_M(s1)	유니코드문자열s1을 멀티바이트코드로 변환한다.
	strcat(s1,s2)	s3 = strcat(s1, s2); // 문자열s1, s2를 결합해서 s3에 리턴한다.
	strcmp(s1,s2)	문자열s1과 s2를 비교한다. 사전적 비교 결과로써 동일할 경우 0를 반환
	strlen(s)	문자열 s의 길이를 얻는다.
	tolower(c)	소문자 바꾸기
	toupper(c)	대문자 바꾸기
	strlwr(s)	소문자열 바꾸기
	strupr(s)	대문자열 바꾸기
	sputc(s,i,c)	s문자열의 i번째 문자를 c로 대체한다.(i가 음수면 끝부터 반대로)
	sgetc(s,i)	s문자열의 i번째 문자를 리턴한다. (i가 음수면 끝부터 반대로)
	ftos(r)	실수r을 문자열로 변환
	itos(i)	정수i를 문자열로 변환
	stof(s)	문자열s를 실수로 변환
	stoi(s)	문자열s를 정수로 변환
	sprintf(fmt,...)	str = sprintf("%d, %8.3f, %s", 100, 3.141592654, "text");
	sgetkeystr(src,key)	src스트링에서 key=str 에 해당하는 str 을 리턴한다.

2.7.3. 포맷문자열

sprintf(fmt, ...), printf(fmt, ...), fprintf(fp, fmt, ...)함수 내부의 인자는 가변인자로써 fmt에 해당하는 문자열에 의해 뒤에 전달될 인자의 수와 형태가 결정됩니다. fmt문자열을 형식문자열 혹은 포맷문자열이라고 부르며 아래와 같은 형식으로 인자를 전달받게 됩니다. (c언어의 printf()문의 포맷문자열과 유사합니다.)

포맷	형식	설명
%f	실수	실수문자열생성
%8.3f	실수	최대 8자리, 소숫점3자리 실수문자열 생성
%-8.3f , %+8.3f	실수	-는 좌측정렬방식을 의미, +는 모든수치에 +/- 부호값 강제표기
%d	정수	부호있는 정수문자열 생성
%6d	정수	최대 6자리 정수문자열 생성
%-6d , %-6d	정수	-는 좌측정렬방식을 의미, +는 모든수치에 +/- 부호값 강제표기
%u	부호없는 정수	부호없는 정수문자열 생성
%x , %X	hex	16진법 표기
%s	문자열	해당 인자를 문자열로 취급하여 해당인자문자열을 삽입
%10s	문자열	최대 10자리 문자열로 삽입
%-10s	문자열	문자열 좌측정렬
%c	문자	해당 인자를 character 코드로 인식하여 문자를 생성

2.7.4. 시스템 메모리의 문자열 처리

일반적인 내부변수의 문자열정보는 문자열테이블에 등록된 번호값입니다. 그러나 시스템 메모리는 전역으로 접근가능한 메모리이며 이곳에 문자열을 저장해서 사용할 경우 대부분 문자내용 자체를 하나씩 기록하는 방식을 취하는 것이 일반적입니다.
시스템 메모리내의 문자열과 문자열 변수간 데이터교환을 위해 STR()함수와 setSTR()함수를 사용합니다.

(1) 문자열변수에 저장된 문자정보를 시스템 메모리에 기록하는 방법

```
var str;
var sysStr[10]:#12000; // 시스템메모리에 80바이트의 문자열기록영역을 사용한다.
str = "Motion Master";  // 변수에 문자열 기록.
setSTR( sysStr, str );  // 시스템메모리에 문자열기록.
```

(2) 시스템 메모리내의 문자열을 문자열변수로 전송하는 방식.

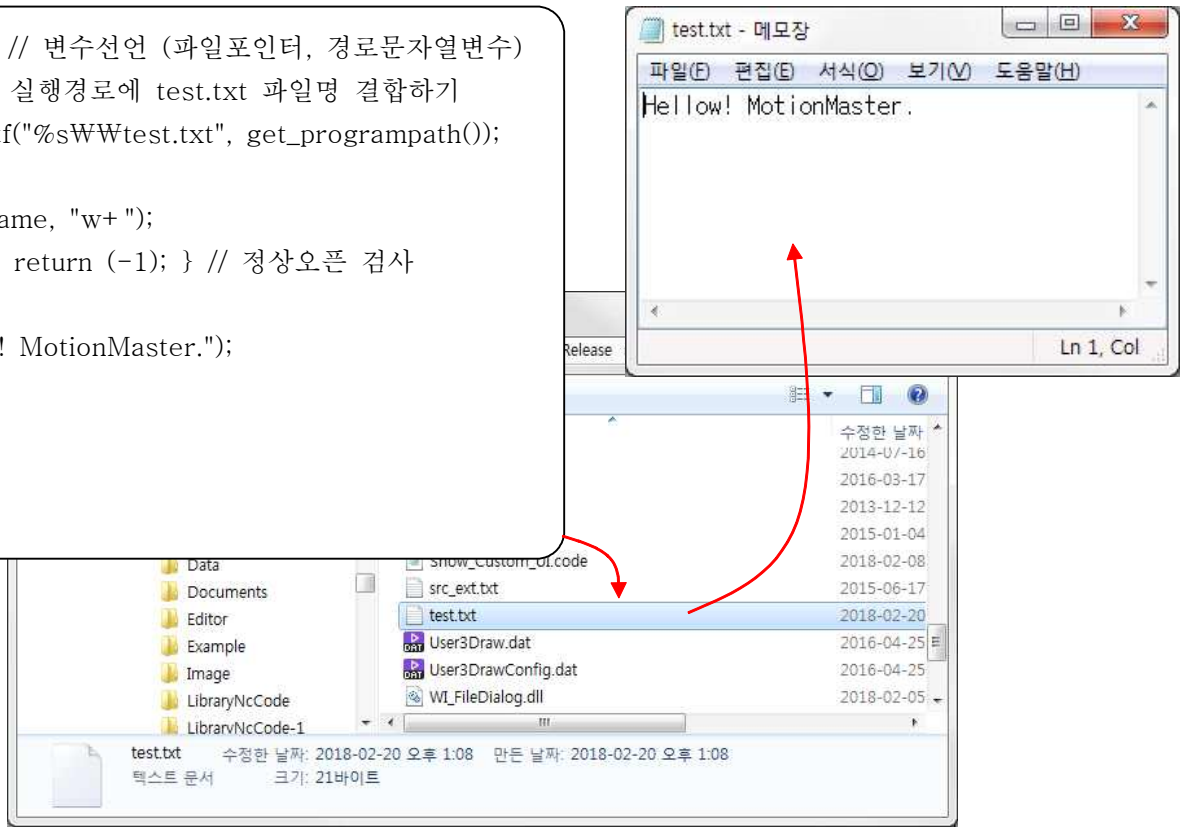
```
var sysStr[10]:#12000; // 시스템메모리에 기록된 문자열 정보
var str;               // 문자열 변수 선언
str = STR(sysStr);     // 시스템메모리에 기록된 문자열을
```

2.8. 파일 처리

파일처리는 사용자가 특정데이터를 저장하고 불러오거나, 특정한 파일의 내용을 분석 처리하는 과정에서 필요합니다. V스크립트는 c언어에서와 같이 파일포인터를 유지하고 관리하는 방식이지만, 파일포인터에 대한 변수형이 없이 단일화된 var 선언으로 사용합니다.

2.8.1. 간단한 파일처리 예제

```
var fp, pathname; // 변수선언 (파일포인터, 경로문자열변수)
// 현재 모션마스터 실행경로에 test.txt 파일명 결합하기
pathname = sprintf("%s₩₩test.txt", get_programpath());
// 파일 열기
fp = fopen( pathname, "w+ ");
if( fp == NULL) { return (-1); } // 정상오픈 검사
// 파일 기록
fprintf(fp, "Hellow! MotionMaster.");
// 파일 닫기
fclose(fp);
// 종료
return(0);
```



프로그램의 전체 흐름은 파일을 열고, 기록하고, 닫는 과정입니다. 파일을 오픈할 때(열 때)는 파일이 정상적으로 오픈되었는지 조사한 후 정상일 때만 다음 작업들이 진행되어야 합니다. "w+" 모드로 오픈할 경우 해당파일을 쓰기용도로 오픈하며 파일이 없을 때는 새로 만들어라는 의미입니다. 아래에 파일을 오픈하는 모드에 대한 문자옵션들을 요약합니다.

r : 읽기 전용으로 열기
w : 쓰기용으로 파일 만들고 열기. 같은 이름의 파일이 있다면 덮어쓰기.
a : 쓰기용으로 파일을 만들고 열기. 단, 같은 이름의 파일이 있다면 그 파일의 끝에 추가하여 쓰기.
r+ : 이미 존재하는 파일을 읽기/쓰기용으로 열기 (업데이트).
w+ : 읽기/쓰기용으로 파일 만들어서 열기, 단, 같은 이름의 파일이 있다면 덮어쓰기.
a+ : 읽기/쓰기용으로 파일 만들어서 열기, 단, 같은 이름의 파일이 있다면 그 파일의 끝에 추가하여 쓰기.
t : 텍스트파일모드(text file mode)로 열기 (삭이 r, w, a, r+, w+, a+와 조합해서 사용)
b : 바이너리파일모드(binary file mode)로 열기 (삭이 r, w, a, r+, w+, a+와 조합해서 사용)

2.8.2. 파일 관리함수들

종류	함수명	설명
	save_datafile(fn,m,s)	fn(파일명)로 지정된 파일에 memory(배열혹은시스템메모리)의 데이터를 size개수만큼 저장합니다.load_datafile()함수와 쌍으로 사용됩니다.
	load_datafile(fn,m,s)	fn(파일명)로 지정된 파일의 내용을 memory(배열혹은시스템메모리)에 size개수만큼 로딩합니다. save_datafile()함수와 쌍으로 사용됩니다.
	load_file_alloc(fn)	fn(파일명)로 지정된 파일의 내용을 내부메모리에 할당받은 포인터를 리턴합니다.
	logfile(title,context)	로그파일에 title문자열과 context문자열의 로그를 기록합니다.
	fopen(fn,m)	fn=파일명, m=오픈모드("r","w","r+","w+"), 반환=파일포인트
	fputc(fp, c)	fp로 지정된 파일에 c문자 기록
	fgetc(fp)	fp로 지정된 파일에 문자얻기
	fputs(fp, s)	fp로 지정된 파일에 문자열쓰기
	fgets(fp)	fp로 지정된 파일에 문자열읽기
	fgetl(fp)	fp로 지정된 파일에 1줄 읽기
	fprintf(fp,fmt,...)	fp로 지정된 포맷의 문자열쓰기
	feof(fp)	파일 끝 검출
	fseek(fp,m,offset)	파일포인터 위치이동
	fsize(fp)	파일크기
	fclose(fp)	파일닥기
	pathfileexist(f)	파일/경로 존재검출
	deletefile(f)	파일삭제
	makedir(d)	폴더생성
	removedir(d)	폴더삭제
	splite_pathname(p,n)	p로 지정된 경로명을 n에 따라서 드라이브명(n=1), 경로명(n=2), 파일명(n=3), 확장자명(n=4), 드라이브+경로명(n=5), 파일명+확장자(n=6)으로 분해.

2.8.3. call()함수를 이용한 사용자 구성파일 관리.

자신만의 머신 응용프로그램으로 만들 경우 다양한 기능들의 셋업 데이터들을 구성파일로 저장하고, 다음 실행 시작 때 읽어들여 메모리를 초기화하는 작업이 자주 발생합니다.
이때 call()함수로써 구성파일을 실행하여 간단하게 구성설정 메모리를 셋업 할 수 있습니다.

(1)먼저 자신의 응용머신의 동작과 관련된 주요한 메모리정보를 정의하는 헤더파일을 만듭니다.
(2)헤더파일에 정의한 다양한 변수들에 값을 설정하는 문자열로 구성된 파일에 저장합니다. (사용자 구성파일저장)
(3) 응용프로그램이 시작되고 리셋단계가 되면 해당 구성파일을 호출(call())하여 실행.
상기 과정으로써 사용자 응용프로그램의 데이터가 초기화 될 수 있습니다. 이 과정을 간단한 예제로써 살펴봅니다.

(1) 자신의 응용프로그램 머신의 주요변수(메모리)가 선언된 헤더파일 만들기 (파일명 예: system.h)

```
var xmin:#10000, ymin:#10001;
var width:#10002, height:#10003;
var title;
```

(2) 사용자 구성파일을 만들기 (파일명 예: config.code)

```
include("system.h");
var fp;
fp = fopen( "config.code", "w+ ");
if( fp == NULL ){ return (-1); }

fprintf(fp, "₩n Config file ");
fprintf(fp, "₩n xmin=%f", xmin);
fprintf(fp, "₩n ymin=%f", ymin);
fprintf(fp, "₩n width=%f", width);
fprintf(fp, "₩n height=%f", height);
fprintf(fp, "Wn title=%s", title);
fclose(fp);
return(0);
```

(3) 구성파일 읽어오기(실행)

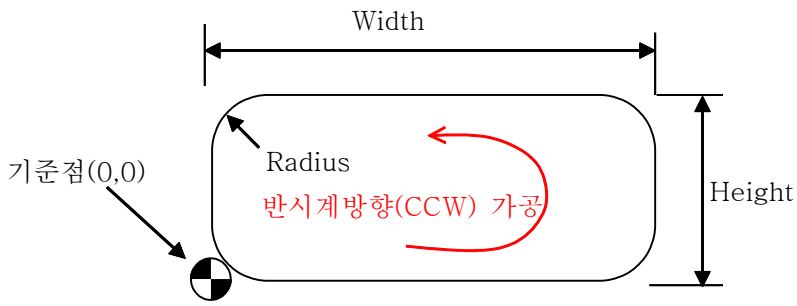
```
include("system.h");
return ( call("config.code") );
```

(4) 리셋단계에서 사용자 구성파일을 읽어오려면
State_Reset_Post.code 파일의
사용자코드영역에
(3) 코드를 삽입합니다.

2.8.4. 자동으로 G코드파일 생성시키기.

특정 데이터값으로 특정형상의 G코드파일을 생성한 후 자동으로 로딩하여 실행하는 코드예제를 살펴봅니다.
먼저 특정데이터를 선언한 system.h 파일을 만들어 봅니다. 각 파일들은 모션마스터실행폴더의 ScriptCode폴더내에 자신만의 코드를 모아두는 임의의 폴더를 생성하거나 UserScriptCode 라는 이미 만들어진 폴더를 이용해서 한곳에 모아서 관리합니다.

<목적형상 >



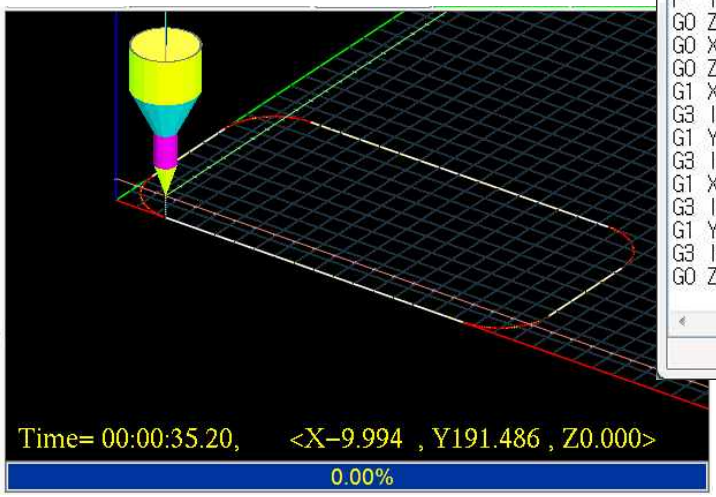
<system.h파일>

```
var Width:#10000, Height:#10001, Radius:#10002; // 10000 번지부터 19999번지까지 사용자시스템에서 사용
var ZSafePos(10), ZFeedPos(0), Feed(1000);
Width = 200; Height= 100; Radius = 25;
```

<generate_NcCode.code 파일>

```
var fp, filename;
filename = sprintf("%s₩₩LibraryNcCode₩₩Test.nc", get_programpath());
fp = fopen( filename, "w+");
if(fp == NULL ) { return (-1); }
fprintf(fp, "₩nG90G21G40G49G15G50G69G40G43");      // 초기화NC코드
fprintf(fp, "₩nG54G52X0Y0Z0A0B0C0");               // G54좌표계사용 및 로컬좌표계클리어
fprintf(fp, "₩nF%-8.1f", Feed);                    // 작업속도설정
fprintf(fp, "₩nG0 Z%-8.3f", ZSafePos);             // Z축 안전위치이동
fprintf(fp, "₩nG0 X%-8.3f Y%-8.3f", Radius, 0.0 );          // 초기위치이동
fprintf(fp, "₩nG0 Z%-8.3f", ZFeedPos);                      // Z축 작업위치이동
fprintf(fp, "₩nG1 X%-8.3f", Width-Radius);
fprintf(fp, "₩nG3 I%-8.3f J%-8.3f X%-8.3f Y%-8.3f", 0, Radius, Width, Radius);
fprintf(fp, "₩nG1 Y%-8.3f", Height-Radius);
fprintf(fp, "₩nG3 I%-8.3f J%-8.3f X%-8.3f Y%-8.3f", -Radius, 0, Width-Radius, Height);
fprintf(fp, "₩nG1 X%-8.3f", Radius);
fprintf(fp, "₩nG3 I%-8.3f J%-8.3f X%-8.3f Y%-8.3f", 0, -Radius, 0, Height-Radius);
fprintf(fp, "₩nG1 Y%-8.3f", Radius);
fprintf(fp, "₩nG3 I%-8.3f J%-8.3f X%-8.3f Y%-8.3f", Radius, 0, Radius, 0);
fprintf(fp, "₩nG0 Z%-8.3f", ZSafePos);             // Z축 안전위치이동
fclose(fp); // 파일닫기

WINC_LOAD( filename, 1); // nc파일을 로딩 후 바로 실행한다. 1이 생략되면 로딩동작만 실행.
return(0);
```

<실행결과>



Test1.nc - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
G90G21G40G49G15G50G69G40G43
G54G52X0Y0Z0A0B0C0
F  1000.0
G0 Z10.000
G0 X25.000    Y0.000
G0 Z0.000
G1 X175.000
G3 I0.000     J25.000   X200.000  Y25.000
G1 Y75.000
G3 I-25.000   J0.000    X175.000  Y100.000
G1 X25.000
G3 I0.000     J-25.000  X0.000    Y75.000
G1 Y25.000
G3 I25.000    J0.000    X25.000   Y0.000
G0 Z10.000
```

Ln 16, Col 13

2.9. 모션컨트롤보드의 입출력 제어용 PLC명령들.

위칸의 모션컨트롤보드(혹은 '모션보드')를 제어하기 위해서 모션보드의 기본적인 특징을 이해해야 합니다.
아래에 V스크립트의 제어입장에서 본 모션보드의 특징을 요약합니다.

(1) 모션보드와 PC는 시리얼통신방식으로 연결되어 있어 초당 전송 가능한 명령이 제한되어 있습니다.
(2) 모션보드는 큐명령과 즉시실행명령이라는 두 가지 방식의 명령을 받습니다.
(3) 큐명령은 모션보드의 명령버퍼에 축적되어 하나씩 순차적이며 연속적으로 실행되는 명령에 유용합니다.
(4) 즉시실행명령은 비상정지 및 비동기적인 입출력동작 등의 명령에 유용합니다.
(5) 즉시실행명령은 큐와 같은 대기구조가 아니므로 명령전송 즉시 실행됩니다. 큐명령에 비해 높은 실행우선순위를 갖습니다.
(6) 즉시실행명령은 현재 모션보드 상태에 따라 실행되지 않고 패스될 수도 있습니다. 그러므로 자신의 원하는 상태가 될 때까지 여러 차례 명령이 전송되어야 될 수도 있습니다.
(7) 즉시실행명령의 전송속도는 통신 연결 속도과 관련되어 있지만, 대체적으로 1~10msec 이상의 여유로 전송하는 것이 바람직합니다. 큐명령과 다른 내부적 데이터흐름에 방해되지 않을 수 있는 전송속도를 유지하는 것이 좋습니다.
(8) 모션보드에서 상태를 입력받는 명령은 통신부하를 주지 않습니다. 모션보드의 상태정보는 엔진구성의 상태타이머에 의해서 자동으로 갱신되므로 별도의 읽기통신명령을 사용하지 않습니다.

모션마스터의 IO 입출력과 관련된 동작은 Background.code 혹은 Background_UI.code 스크립트파일에 작성되는 것이 일반적이며, 이들 파일들의 자동실행 반복속도는 초당 수십회에 불과함으로 명령전송흐름의 과부하현상은 거의 발생되지 않습니다.

2.9.1. 큐명령과 즉시실행명령

CNC 자동동작을 시작하면 NC코드는 즉시 번역되어 모션보드의 큐명령방식으로 전송됩니다.
즉 큐명령은 CNC 자동동작의 연속보간 등을 위한 전용의 명령방식으로 생각할 수 있습니다.
대부분 사용자가 사용하는 입출력명령들은 즉시실행명령체계로 전송되어져야만 CNC자동동작과 충돌이 없습니다.

즉시실행명령들은은 아래의 함수구조로 사용합니다.

```
WINC_ENTER_IMCMD();  // 즉시실행명령의 시작을 알리는 함수.

        .... ;              // 모션보드 즉시실행명령들 나열.

WINC_LEAVE_IMCMD(); // 즉시실행명령의 끝을 알리는 함수.
```

큐명령들은 아래의 함수구조로 사용합니다.

```
WINC_ENTER_QUEUECMD();  // 큐명령의 시작을 알리는 함수.

        .... ;                      // 모션보드 큐명령들 나열.

WINC_LEAVE_QUEUECMD(); // 큐명령의 끝을 알리는 함수.
```

2.9.2. 범용 입출력 함수들

V스크립트언어는 WINC_ 로 시작하지 않는 기본적인 입출력함수들을 제공합니다. 아래의 함수는 자신이 적용되는 장비에 대한 범용적인 입출력 함수이므로 쉽고 일관되게 입출력포트에 접근할 수 있습니다.

종류	함수명	설명
	getPout(n)	n핀번호에 해당하는 출력핀의 상태(0,1)를 읽어들입니다.
	getPin(n)	n핀번호에 해당하는 입력핀의 상태(0,1)를 읽어들입니다.
	setPout(n)	n핀번호에 해당하는 출력핀의 상태를 1로 설정합니다.
	clrPout(n)	n핀번호에 해당하는 출력핀의 상태를 0으로 설정합니다.
	togPout(n)	n핀번호에 해당하는 출력핀의 상태를 반전시킵니다.
	putPout(n, val)	n핀번호에 해당하는 출력핀의 상태를 val값(0,1)으로 설정합니다.
	getPwout(n)	n포트번호에 해당하는 출력포트의 워드상태를 읽어들입니다.
	getPwin(n)	n포트번호에 해당하는 입력포트의 워드상태를 읽어들입니다.
	putPwout(n, val)	n포트번호에 해당하는 출력포트의 워드상태를 val값으로 설정합니다.
	setPwm(ch, p, d)	ch채널에 해당하는 Pwm을 주기 p, 듀티 d값으로 설정합니다.
	getPwm(ch, n)	ch채널에 해당하는 Pwm의 상태값을 읽습니다. (n=0:주기, n=1:듀티)
	getAdc(ch)	ch채널에 해당하는 Adc값을 읽습니다.
	getEnc(ch)	ch채널에 해당하는 엔코더값을 읽습니다.
	getCnt(ch)	ch채널에 해당하는 카운터값을 읽습니다.

핀 혹은 포트값의 상태는 시스템메모리의 특정영역에 맵핑되어 있지만, 상기함수를 통해서도 읽어들 일 수 있습니다. 핀, 포트, 혹은 채널 값은 아래의 범위에 따라 모션마스터와 연결된 개별 장치들을 자동으로 선택하게 됩니다.

범위	함수명	설명
0~99	모션컨트롤메인보드	0~31까지 사용가능
100~199	PIX48 #1보드	100이면 0번 핀/포트/채널, 199이면 99번 핀/포트/채널 로 지정
200~299	PIX48 #2보드	200이면 0번 핀/포트/채널, 299이면 99번 핀/포트/채널 로 지정
이후 예약		

getPin()함수와 같은 상태입력함수들은 바로 사용될 수 있지만, 출력함수의 경우 큐명령으로 처리할 지, 즉시실행명령으로 처리할 지를 결정해야 합니다. 큐명령으로 전송할 경우, 현재 명령큐에 다수의 처리되지 않은 명령들이 있다면 이들 명령이 전부 처리되고 난 후에야 실행되게 됩니다. 그러므로 일반적으로 아래와 같이 즉시실행명령을 사용합니다.

```
WINC_ENTER_IMCMD();
        setPout(0);
WINC_LEAVE_IMCMD();
```

2.9.3. Background.code 에서 사용되는 편리한 PLC 명령 함수들

Backgournd.code 스크립트 파일내에서 입출력제어 프로그램을 작성할 때 범용입출력 함수보다 편리한 PLC함수명령들을 사용할 수 있습니다.
이들 명령은 자동으로 즉시실행명령으로 처리되고, 동일한 상태의 출력이 여러 번 중복되더라도 단 한번만 출력되도록 조정되어 다수의 출력명령에서도 모션보드 통신부하량에 영향을 주지 않습니다.

함수이름에서 X를 포함하면 입력포트와 관련되고 Y를 포함하면 출력포트와 관련된 입출력함수입니다.
그외 100개의 타이머와 PLC의 M메모리와 유사한 1비트 메모리 등을 사용할 수 있으며, 입력신호의 엣지(상승엣지, 하강엣지)를 검출하여 동작하는 명령들도 포함됩니다. 아래에 PLC명령함수들을 요약합니다.

(1) PLC 전체 동작제어

종류	함수명	설명
실행속도제어	looptime(sec)	Background.code실행속도를 sec초단위로 설정합니다.
버퍼링된 입출력	read_port()	전체입력을 읽어서 버퍼링합니다. 일반적으로 Background.code 의 선두에서 호출합니다.
	write_port()	버퍼링된 출력을 장치로 전송합니다. 일반적으로 Background.code 의 마지막에 호출합니다.
타이머운영	run_T()	타이머 동작을 활성화 합니다. 일반적으로 Background.code 의 마지막에 호출합니다. PLC타이머는 100개까지 사용가능합니다.
M메모리운영	run_M()	M메모리동작을 활성화합니다. 일반적으로 Background.code 의 마지막에 호출합니다.

상기 명령들은 Background.code 파일의 시작과 끝에 작성되어 plc명령함수들이 정상적으로 동작할 수 있도록 해줍니다. 아래에 Background.code 함수의 전체적인 코드흐름을 상기 함수들 입장에서 기술해봅니다.
- Background.code 파일 내용요약-

```
// 시작코드
read_port(); // 전체 입력버퍼링
looptime(0.05); // 실행시간제어

// 사용자 코드영역
...
...
...

// 종료코드
run_T(); // 타이머 운영제어
run_M(); // M메모리 운영제어
write_port(); // 전체 출력버퍼링
return(0);
```

(2) 입출력 명령

종류	함수명	설명
	X(i)	i번째 입력값을 얻습니다.
	pX(i)	i번째 입력의 상승에지를 검출합니다.
	nX(i)	i번째 입력의 하강에지를 검출합니다.
	cX(i)	i번째 입력의 변화를 검출합니다.
	Y(i)	i번째 출력값을 얻습니다.
	pY(i)	i번째 출력의 상승에지를 검출합니다.
	nY(i)	i번째 출력의 하강에지를 검출합니다.
	cY(i)	i번째 출력의 변화를 검출합니다.
	setY(i)	i번째 출력값을 1로 설정합니다.
	clrY(i)	i번째 출력값을 0으로 설정합니다.
	togY(i)	i번째 출력값을 토글합니다.
	putY(i,v)	i번째 출력값으로 v값으로 설정합니다.
일괄포트입출력 확장	Xw(i) / Yw(i)	w 접미사는 word라는 의미로 하나이 신호핀이 아니라 포트입출력 전체를 의미합니다. (Xw(), Yw(), setYw(,), clrYw(,), putYw(,) 함수로 확장적용)

포트번호에 대한 적용은 범용입출력함수와 동일합니다.

(3) 타이머명령

종류	함수명	설명
	startT(i, time)	i번째 타이머를 time설정값으로 시작합니다.
	setT(i, time)	i번째 타이머의 설정시간을 time으로 설정합니다.
	getT(i)	i번째 타이머의 현재시간값을 얻습니다.
	stopT(i)	i번째 타이머의 동작을 정지시킵니다.
	restartT(i)	i번째 타이머의 동작을 재개시킵니다.
	resetT(i)	i번째 타이머를 리셋합니다.
	TON(i)	i번째 타이머의 ON동작결과(설정시간경과완료상태)를 얻습니다.
	TOFF(i)	i번째 타이머의 OFF동작결과(설정시간경과비완료상태)를 얻습니다.

최대 100개의 타이머(0~99번)를 사용할 수 있습니다. PLC타이머 외에 시스템메모리에서 제공하는 타이머기능과 타임블링크 플래그의 기능들을 같이 사용하면 편리할 수 있습니다. 시스템 메모리의 타이머플래그편을 참조할 수 있으며 #30400 ~#30499까지의 영역입니다. PLC타이머의 0번 타이머는 Background_Setup.code 내에 1초 값으로 설정되어 자동으로 동작되고 있습니다.

(3) M메모리 조작명령

종류	함수명	설명
	M(i)	i번째 M메모리값을 얻습니다.
	setM(i)	i번째 M메모리값을 1로 설정합니다.
	clrM(i)	i번째 M메모리값을 0으로 설정합니다.
	togM(i)	i번째 M메모리값을 토글합니다.
	putM(i,v)	i번째 M메모리값을 v값으로 설정합니다.
	pM(i)	I번째 메모리값의 상승에지를 검출합니다.
	nM(i)	I번째 메모리값의 하강에지를 검출합니다.
	cM(i)	I번째 메모리값의 변화를 검출합니다.

M메모리는 최대 8192개(0~8191)까지 사용가능합니다. (총 1k바이트)

2.10. 모션마스터 제어 전용함수들

WINC_ 로 시작되는 함수는 모션마스터의 제어동작과 직접적으로 관련된 함수들입니다.

아래에 간단한 예로써 동작을 설명합니다.
- WINC_START() 함수를 호출하면, 현재 로딩되어 있는 NC코드파일의 실행하게 됩니다.
- WINC_PAUSE() 함수를 호출하면, 현재 동작중인 모션이 일시정지하게 됩니다.
- WINC_ESTOP() 함수를 호출하면, 비상정지를 발생하게 됩니다.

모션마스터 제어 전용함수 수가 많으므로 몇 가지 그룹으로 요약해 봅니다.
(V스크립트 버전에 따라 추후 추가될 수 있습니다.)

2.10.1. 기초 변환

모션마스터의 축지정방식은 축선택값과 축번호값 두가지 방식으로 사용될 수 있습니다.
축선택값은 총8비트로써 LSB 0번비트부터 X축 Y,Z,A,B,C,S,S2축으로 각 비트가 1일때 선택되어졌음을 의미합니다.
축번호값은 0이 X축 1이 Y축 ... S축은 6이 됩니다. 아래의 함수는 축번호값을 축선택값으로 축선택값을 축번호값으로 각각 바꾸어 주는 함수입니다.

종류	함수명	설명
축변환	WINC_TO_AXIS(n)	축 인덱스값 n을 축 지정값으로 변환합니다. n=0 이면 X축으로 0x01 값을 리턴하며, n=1 이면 Y축으로 0x02 값을 리턴합니다. 리턴값을 c언어방식으로 표현하면 return (0x01 << n); 이 됩니다.
	WINC_TO_AXISNO(axis)	axis로 설정된 축 지정값을 축인덱스값으로 변환시킵니다. axis=0x01 이면 0을 리턴하며, axis=0x02 이면 1을 리턴합니다. LSB로부터 첫번째로 만나는 비트1의 위치를 리턴합니다.

2.10.2. 명령실행 방식제어

종류	함수명	설명
즉시실행명령	WINC_ENTER_IMCMD()	이후 명령들을 즉시실행명령으로 처리합니다.
	WINC_LEAVE_IMCMD()	즉시실행명령방식을 종료합니다.
큐실행명령	WINC_ENTER_QUEUECMD()	이후 명령들을 큐실행명령으로 처리합니다.
	WINC_LEAVE_QUEUECMD()	큐실행명령방식을 종료합니다.

기본적으로 모션보드명령은 큐명령실행으로 처리됩니다. 즉시실행명령으로 사용할 경우 아래의 순서로 사용해야 합니다.

```
WINC_ENTER_IMCMD() // 즉시실행명령을 시작합니다.
.... // 명령을 기록합니다.
WINC_LEAVE_IMCMD() // 즉시실행명령을 종료합니다.
```

2.10.3. 입출력 제어

아래의 모션마스터 입출력제어함수는 V스크립트의 범용입출력제어함수와 중첩됩니다.
가능하면 V스크립트의 범용입출력함수를 사용하는 것을 추천합니다.

<일반입출력>

종류	함수명	설명
GIO 입출력	WINC_SET_GIO(p)	출력포트 p를 SET 합니다. (p는 16진수)
	WINC_CLR_GIO(p)	출력포트 p를 CLR 합니다. (p는 16진수)
	WINC_GET_GIO(a)	입력포트 값을 리턴합니다.(a=0x01, 리턴값은 16진수)
EIO 입출력	WINC_ENABLE_EIO(enable)	enable = 0 이면 확장IO 연결을 DISABLE합니다. enable = 1 이면 확장IO 연결을 ENABLE합니다.
	WINC_GET_EIO(ch)	확장IO가 ENABLE되었을 때 연결된 IO보드채널의 입력포트 값을 리턴합니다.
	WINC_SET_EIO(ch)	확장IO가 ENABLE되었을 때 연결된 IO보드채널에 출력포트 값을 설정합니다.

<카운터, 엔코더>

종류	함수명	설명
카운터(업)	WINC_GET_CNT(ch)	ch 채널의 카운터값을 읽습니다.
	WINC_SET_CNT(ch, value)	ch 채널의 카운터값을 value값을 설정합니다.
엔코더	WINC_ENABLE_ENC(ch,enable)	ch = { 0, 1 } 범위로 GIO의 엔코더 채널 enable = 0 이면 비활성화 1 이면 활성화 됩니다.
	WINC_GET_ENC(ch)	ch = {0,1} 채널번호 해당 채널의 엔코더카운터값을 읽어서 리턴합니다.

<PWM 출력 제어>

종류	함수명	설명
PWM	WINC_SET_PWM(ch, period, duty)	ch = {0, 1} 채널번호 period = 주기결정 카운터값 duty = 출력비 결정 카운터값 x6v3엔진기준> 실제주기 = 20kHz / period 해당 채널의 pwm파형을 출력합니다.

<ADC 입력>

종류	함수명	설명
ADC	WINC_GET_ADC(ch)	ch = {0,1,2,3} 채널번호 해당 채널의 adc값을 읽어서 리턴합니다.

2.10.4. 서보 입출력 제어

종류	함수명	설명
서보온	WINC_SERVO_ON(axis,val[])	axis축지정, val[]는 최대 8배열로 해당축 서보의 on/off(0/1)값 지정합니다. (X6V3엔진 모션컨트롤러는 통합된 하나의 SERVO ON신호출력만 가능합니다. 이때 축지정은 0xff로 전체지정합니다.)
서보리셋	WINC_SERVO_RESET(axis,val[])	axis축지정, val[]는 최대 8배열로 해당축 서보의 on/off(0/1)값 지정합니다. (X6V3엔진 모션컨트롤러는 통합된 하나의 SERVO RESET신호출력만 가능합니다. 이때 축지정은 0xff로 전체지정합니다.)

2.10.5. 주동작 제어

종류	함수명	설명
자동시작	WINC_START()	자동가공을 시작합니다.
일시정지	WINC_PAUSE()	일시정지시킵니다.
동작취소	WINC_CANCEL()	일시정지혹은 정지된 동작을 완전히 취소합니다.
자동종료	WINC_END()	자동가공을 종료합니다.
비상정지	WINC_ESTOP()	비상정지시킵니다.
원점복귀	WINC_HOME(axis)	axis=0이면 전체 시퀀스 원점복귀를 실행합니다. 자동동작 중일때는 허용되지 않습니다.
리셋	WINC_RESET()	소프트리셋동작을 수행합니다.
종료	WINC_EXIT()	프로그램을 종료시킵니다.
알람 리셋	WINC_ALARM_RESET()	알람상태를 리셋시킵니다.
명령큐클리어	WINC_CLR_QUEUE()	현재 실행대기 중인 명령큐를 삭제합니다.
머신록	WINC_MACHLOCK(b)	b=1 : 머신록 동작으로 진입합니다. b=0 : 머신록 동작을 해제합니다. (반드시 1과 0동작을 한쌍으로 사용하여야 합니다.)
싱글블록	WINC_SINGLEBLOCK(b)	b=1 : 싱글블록 설정입니다. b=0 : 싱글블록 해제입니다.
블록스킵	WINC_BLOCKSKIP(b)	b=1 : 블록스킵을 설정합니다. b=0 : 블록스킵을 해제합니다.
옵셔널스톱	WINC_M01STOP(b)	b=1 : 옵셔널스톱을 설정합니다. b=0 : 옵셔널스톱을 해제합니다.
알람 해제	WINC_RELEASE_ALARM(b)	b=1 : 현재 알람상태이면 일시적으로 알람을 해제시킵니다. b=0 : 해제된 알람에서 원래상태로 복귀합니다. (반드시 1과 0동작을 한쌍으로 사용하여야 합니다.)

2.10.6. 구성파일과 자동가공파일 로딩, 저장

종류	함수명	설명
구성설정저장	WINC_SAVE_CONFIG(p, b)	지정된 파일(p)에 현재구성설정을 저장합니다. b=1 이면 종료시점에서 저장되어야 하는 기본적인 구성들만 저장되며, b=0 이면 모든 정보를 저장합니다. 지정된 파일경로 p가 “”로 아무런 경로가 없다면 내정된 구성설정파일에 덮어쓰게 됩니다.
구성설정로딩	WINC_LOAD_CONFIG(fn)	fn은 경로가 포함된 구성파일명 fn이 “”으로 지정이 없으면 구성파일 리로딩기능
자동가공파일 로딩	WINC_LOAD(fn); WINC_LOAD(fn, run);	fn은 자동동작용 nc파일명(경로포함) fn이 “”으로 지정이 없으면 새로운 파일로딩 기능 run = 1 이면 로딩 후 바로 가공시작. (fn 지정이 있고 파일이 존재할 경우에만)
리로딩	WINC_RELOAD();	

2.10.7. MDI 실행요청

종류	함수명	설명
실행요청	WINC_MDI_FILE("motion.nc");	"motion.nc" 파일 실행을 요청합니다. (자동동작상태가 아닐때)
	WINC_MDI("G21G90");	G코드 문자열 실행을 요청합니다. (자동동작상태가 아닐때)

2.10.8. 피드오버라이딩

종류	함수명	설명
	WINC_FEED_OVERRIDE(p)	0~100 지정
	WINC_G00_OVERRIDE(p)	0~100 지정
	WINC_SPINDLE_OVERRIDE(p)	0~100 지정

2.10.9. 공구번호지정

종류	함수명	설명
	WINC_SET_TOOLNO(curT, preT);	curT = -1 이면 지정변화 없음 preT = -1 이면 자동지정 preT = -2 이면 지정변화 없음

2.10.10. MPG 선택

종류	함수명	설명
	WINC_MPG(n)	n=1,2,3 에 해당하는 mpg를 선택합니다. n=0이면 현재 mpg번호를 리턴합니다.

2.10.11. 좌표계

종류	함수명	설명
좌표계	WINC_GET_WORKCOORD_NO()	현재좌표계 방식을 얻습니다. 53=G53, 54=G54...
	WINC_SET_WORKCOORD_NO(n)	n으로 지정된 방식으로 현재 좌표계를 설정합니다. G53: n=53, G54: n=54...
	WINC_SET_WORKCOORD(axis,pos[])	현재좌표계의 좌표를 pos로 설정합니다. axis = 0 이면 전체 워크좌표를 설정합니다. * 완전히 정지한 상태에서 사용해야하며, 로컬좌표,기계좌표를 변화시키지 않으며, G54 등의 공작물좌표계의 원점을 바꿉니다. 이때 회전, 스케일 등은 적용되지 않으므로 반드시 이들 옵션들을 해제한 후 사용합니다.
	WINC_GET_MACHPOS(pos[])	현재 기계좌표값을 pos배열에 기록합니다.
	WINC_SET_MACHPOS(axis, pos[])	pos배열에 기록된 값으로 현재좌표값을 설정합니다. axis=0이면 전체 축의 좌표값을 설정합니다.
	WINC_UPDATE_NCPOS()	현재 기계위치값으로 NC위치를 새롭게 갱신합니다.
	WINC_TO_NCPOS(mPos[], ncPos[])	mPos[]의 기계위치값을 NC위치값으로 바꾸어 ncPos[]에 저장합니다.
	WINC_TO_NCPOS_COMPH (mPos[], ncPos[])	mPos[]의 기계위치값을 공구장보정이 적용된 NC위치값으로 바꾸어 ncPos[]에 저장합니다.
	WINC_TO_MACH(ncPos[], mPos[]);	ncPos[]의 NC위치값을 기계위치값으로 바꾸어 mPos[]에 저장합니다.

2.10.12. 조그 동작

종류	함수명	설명
	WINC_JOG_ENABLE(b)	조그/MPG동작 여부를 설정합니다.
	WINC_JOG_MODE(m)	m=0 : 스텝조그동작 m=1 : 연속조그동작
	WINC_JOG_MOVE(axis,dir)	axis : 축지정값 dir = 0 : 축정지 dir = 1 : 증가방향 동작 dir = -1 : 감소방향 동작
	WINC_JOG_MASTER_SPEED(p)	p = 0~100% 값의 비율의 조그 한계동작속도 비율설정
	WINC_JOG_SPEED(p)	p = 0~100% 값의 비율로 조그 메인동작속도 비율설정
	WINC_JOG_AXIS_SPEED(p)	p = 0~100% 값의 비율의 조그 축별동작속도 비율설정
	WINC_JOG_MAIN_SPEED_AXIS(axis)	메인동작속도에 영향 받는 축 지정
	WINC_JOG_MULTIPLIER(m)	m = 1,10,100,1000 에서 배율선택

2.10.13. 시스템 메모리 액세스

종류	함수명	설명
	WINC_PUT_DATA_REAL(i,f)	시스템 메모리 I번지에 실수값 f를 기록합니다.
	WINC_GET_DATA_REAL(i,f)	시스템 메모리 I번지내용을 실수값으로 읽습니다.
	WINC_PUT_DATA_INT(i,d)	시스템 메모리 I번지에 정수값 d를 기록합니다.
	WINC_GET_DATA_INT(i)	시스템 메모리 I번지내용을 정수값으로 읽습니다.
	WINC_PUT_DATA_HEX(i,h)	시스템 메모리 I번지에 부호없는정수값 h를 기록합니다.
	WINC_GET_DATA_HEX(i)	시스템 메모리 I번지내용을 부호없는저수값으로 읽습니다.
	WINC_PUT_DATA_BOOL(i,b)	시스템 메모리 I번지에 논리값 b를 기록합니다.
	WINC_GET_DATA_BOOL(i)	시스템 메모리 I번지내용을 논리값으로 읽습니다.
문자열	WINC_PUT_DATA_STR(i,"str")	시스템 메모리 I번지에 문자열 str을 기록합니다.
문자열	WINC_GET_DATA_STR(i)	시스템 메모리 I번지내용을 문자열값으로 읽습니다.
	WINC_PUT_DATA_BIT(i, n, b);	시스템 메모리 i번지의 n번째 비트를 b값으로 기록합니다.
	WINC_GET_DATA_BIT(i,n);	시스템 메모리 i번지의 n번째 비트를 읽습니다.

2.10.14. 펑션키 호출

종류	함수명	설명
	WINC_FUNCKEY(n)	n=1 이면 F1키 동작을 실행합니다. ... n=8 이면 F8키 동작을 실행합니다.

2.10.15. 축지정과 축교환

종류	함수명	설명
축지정	WINC_ASSIGN_AXIS(ncaxis, bdaxis)	NC의 axis축을 모션보드의 축 bdaxis로 지정합니다.
축교환	WINC_EXCHANGE_AXIS(axis1, axis2)	axis1축과 axis2축을 서로 바꿉니다.

2.10.16. 테이블 등록

종류	함수명	설명
워크테이블 등록	WINC_REGISTER_WORKTABLE(n,a,p)	워크번호 n, 축 a, 위치배열 p (G52=52, G53=53, G54=54, G55=55...G59=59, G54.00=1(혹은 5400), G54.99=10(혹은 5499))
매크로 테이블 등록	WINC_REGISTER_MACROTABLE(n,v,m,p)	매크로 테이블의 n번째 위치에 유효성여부(v), 매크로이름(m), 파일경로(p) 정보를 등록합니다.
자동프로그램 테이블 등록	WINC_REGISTER_AUTOTABLE(n,t,p,s,c,g)	자동프로그램테이블의 n번째 위치에 프로그램타이틀(t), 파일경로(p), 셋카운터(s), 현재카운터(c), 불량카운터(g) 정보를 등록합니다.

2.10.17. 다양한 다이얼로그화면 호출

종류	함수명	설명
다이얼로그	WINC_DIALOG(n) WINC_DIALOG(n, m)	모션마스터에서 사용하는 다이얼로그 생성 n=0 : 데이터 관리 테이블 다이얼로그 n=1 : 시스템 메모리 테이블 다이얼로그 n=2 : 홈위치 관리 테이블 다이얼로그 n=3 : 워크 테이블 다이얼로그 n=4 : 공구 테이블 다이얼로그 n=5 : 매크로 테이블 다이얼로그 n=6 : 자동프로그램 테이블 다이얼로그 n=8 : 연결구성 초기화 다이얼로그 n=9 : 알람리스트 다이얼로그 n=10: 네트워크 통신설정 오픈 n=11: 기계안내 다이얼로그 n=12: 가공파일 오픈 다이얼로그 m이 생략되면 SendMessage방식으로 호출 m이 1이면 PostMessage방식으로 호출

2.10.18. WINC_CMD() 통합명령함수

WINC_CMC(cmd, para1, ...) 의 형식으로 cmd는 명령문자열, para1, para2, ... 등은
가변인수로써 문자열이 될 수도 있으며 수치값이 될 수 도 있습니다. 다소 명령군이 많아서 그룹으로 나누어서 요약합니다.

그룹	설명	
#1	명령 문자열	설명
	“reset", "start", "pause", "cancel", "home", "alarmreset", "alarmrelease" "machlock", "estop"“reload", "exit"	단일명령으로 이름그대로의 동작을 실행합니다. (별도의 파라메터는 없음)
	“singleblock", "optionalstop", "blockskip"	토글명령으로 명령실행때마다 반전동작을 실행합니다. (별도의 파라메터는 없음)
	“load"	para1은 경로를 포함한 nc가공파일이름의 문자열입니다. para1이 없다면 가공파일 오픈다이얼로그가 생성됩니다.
	“enable_auto","enable_jog", "enable_mpg","enable_mdi", "enable_joystick", "enable_edit","enable_home"	para1은 문자열수식으로써 수식의 결과값이 0,1,-1 이어야합니다. 0이면 disable, 1이면 enable, -1이면 toggle동작을 실행합니다. 예> WINC_CMD("enable_mdi", "1");
	"home_x", "home_y", ... "togo_x0", "goto_y0", ...	개별축 원점복귀와 영점이동명령입니다. (파라메터없음)
	"togo_x", "togo_y", ...	MDI의 G0 명령을 요청합니다. para1은 문자열로써 각 축의 위치정보입니다. 예> WINC_CMD("togo_x","123.45");
	"spindle_on", "spindle_off" "spindle_cw", "spindle_ccw"	스핀들 회전명령으로 별도의 파라메터 없음.
	"rpm", "rpm+", "rpm-", "rpm_override", "rpm_override+","rpm_override-"	스핀들 rpm회전값을 설정합니다. para1은 각 설정값에 해당하는 문자열을 전달합니다.
	"feed", "feed+", "feed-", "feed_override", "feed_override+","feed_override-"	피드, 피드오버라이딩을 설정합니다. para1은 각 설정값에 해당하는 문자열을 전달합니다.
	"m8","m9", "m8_toggle", "m10","m11", "m10_toggle", "m14","m15", "m14_toggle", "m38","m39", "m38_toggle", "m68","m69", "m68_toggle",	기본 m코드 실행을 요청합니다. 별도의 파라메터는 없습니다.
	“setgout","clrgout","togglegout", "setgout_moment", "clrgout_moment" "pwm"	IO핀 상태출력명령 pwm명령만 para1, para2, para3의 수치값문자열이 필요합니다. para1문자열은 채널번호값 para2문자열은 주기값, para3문자열은 듀티값입니다.
	"mdi"	

그룹	설명	
#2	명령 문자열	설명
	"jog_step", "jog_continue", "jog_mode_toggle"	조그모드를 설정합니다.
	“jog_speed", "jog_speed+", "jog_speed-",	조그 스피드를 설정합니다. para1은 0~100이내의 수치문자열입니다.
	"jog_enable_probe",	조그 프로브동작을 설정합니다. para1은 0혹은 1의 수치문자열입니다.
	"jog_x1", "jog_x10", "jog_x100", "jog_x1000",	조그 동작 배율값을 설정합니다.
	"jog_x+", "jog_x-", "jog_xstop", "jog_y+", "jog_y-", "jog_ystop", ...	조그 축동작을 지시합니다.
	"work_x", "work_y", "work_z", ...	현재워크좌표값을 설정합니다. para1은 좌표값을 나타내는 문자열입니다.
	"work_0"	현재워크좌표값을 0으로 설정합니다. para1은 0을 만들고자하는 축선택값 문자열입니다.
	"work_x0", "work_y0", "work_z0",... "work_all0"	각 축별 혹은 전체 워크좌표값 0을 설정합니다.
	"f1", "f2", "f3",..."f16"	펑션키 f1 ~ f16을 실행합니다.
	"datatable", "sysmemory", "hometable", "worktable", "tooltable", "macrotable", "autotable","dialog_link", "alarmlist"	해당 이름을 테이블 다이얼로그가 생성됩니다.
	"color_g0", "color_g1", "color_g2", "color_g3",	g0,g1,g2,g3에 해당하는 칼라를 선택합니다. para1은 칼라값에 대한 문자열입니다.

2.11. 시리얼 (LAN) 통신함수

단순한 시리얼 통신은 background.code 내에서 구현할 수 있습니다. 아래에 간단한 시리얼(혹은 이더넷) 통신을 위한 함수들을 요약합니다. 보다 복잡한 프로토콜을 갖는 통신은 별도의 DLL 라이브러리를 만들어서 결합하는 것이 바람직합니다.

함수명	설명

open_com(p1, b);	open_com()함수는 시리얼과 이더넷 통신 모두에 사용될 수 있습니다.
	시리얼 통신의 경우 첫번째인자가 시리얼통신포트번호가 되며 이더넷일 경우 10000번으로 지정합니다.
	두번째 인자는 통신속도 수치값이나 혹은 이더넷의 오픈정보의 문자열값입니다.
	실패하면 -1이 리턴됩니다.
	예1> 시리얼통신
	open_com(5, 115200); // 5번포트를 115200의 속도로 open한다.
	예2> 이더넷통신
	open_com(10000, "ip=192.168.0.52, port=2000, time=10000, reconnect=0");

close_com(n);	통신포트를 닫습니다. n은 통신포트번호입니다.
isempty_com(n);	n통신포트의 수신문자가 비어있으며 참(1), 아니면 거짓(0)을 리턴합니다.
endoftx_com(n)	n통신포트로 전송이 완료되었는지 확인합니다.
getch_com(n)	n통신포트에서 문자하나를 받아옵니다.
getdump_com(n,buf,size)	n통신포트에서 buf[]배열에 최대 size만큼의 데이터를 받아옵니다. 리턴값은 최대 받은 문자(바이트) 수입니다.
putch_com(n,c)	n통신포트에서 문자하나를 전송합니다.
putstr_com(n,str)	n통신포트에서 문자열을 전송합니다.
cprintf(n,fmtstr,...)	n통신포트에서 포맷된 문자열을 전송합니다.

```
/*-------------------------------------------------------------------------------------
        Background.code 파일에서 실행되는 시리얼 통신 및 이더넷 통신테스트 예제
-------------------------------------------------------------------------------------*/
svar stateCom(0), ComNo(8);
var datablock[100]:#50;

// 포트열기
if( stateCom == 0 && #30170 >= 60)
{
    // 이더넷통신예> ComNo=open_com(10000, "ip=192.168.0.52, port=2000, time=10000, reconnect=0");
    ComNo = open_com(ComNo, 115200); // 시리얼통신예

    if( ComNo < 0 ) { statCom = 900; } // 종료
    else { stateCom = 20; } // 정상
}
// 데이터 전송
else if( stateCom == 20 )
{
        putstr_com(ComNo, "₩r Hellow! 0123456789 ");
        stateCom = 40;
}
// 데이터 수신
else if( stateCom == 40 )
{
        svar rxState(0), rxCnt(0), rxCntMax(0);
        var ch;

        // 수신프로토콜
        while( !isempty_com(ComNo) )
        {
                ch = getch_com(ComNo);

                if( rxState == 0 ) // 수신 시작 문자확인 (0X02)
                {
                        if( ch == 0X02 ) { rxState = 10 }
                }
                else if( rxState == 10 ) // 수신 데이터 길이
                {
                        rxCntMax = ch;  rxCnt = 0; rxState = 20;
                }
                else if( rxState == 20 )
                {
                        datablock[rxCnt] = ch; rxCnt++;
                        if( rxCnt >= rxCntMax ) { rxState = 30; }
                }
                else if( rxState == 30 ) // 수신 종료 문자확인 (0X03)
                {
                        if( ch == 0X03 ) { rxState = 0;  stateCom=50; }
                }
        }
}
// 포트 닫기
else if( stateCom == 50 ) { close_com(ComNo); stateCom = 900; }
else if( stateCom == 900 ) { } // 상태종료

// 모션마스터 리셋 / 종료시 포트강제 종료
if( (#30185 && stateCom > 10) || #30175 ) { close_com(ComNo); }
```

2.12. 자료형 변환 함수들

V스크립트의 변수의 기본자료형은 8바이트 double(배정도실수)형 데이터입니다. 그러나 시스템메모리의 내용은 모두가 double형이 아니기 때문에 자료형변환이 필요합니다. 이때 필요한 함수가 자료형 변환함수입니다.
소문자 표기이며 INT()함수가 있으면 그 반대의 toINT()함수가 있어 쌍을 이룹니다.

INT(a)함수는 a라는 값이 32비트 정수데이터형이므로 v스크립트자료형(배정도실수형)으로 바꾸라는 의미입니다.
toINT(b)함수는 b라는 v스크립트자료형(배정도실수형)을 32비트 정수데이터형으로 바꾸라는 의미입니다.

종류	함수명	설명
자료형변환	INT(a) / toINT(b)	32비트 정수변환
	UINT(a) / toUINT(b)	32비트 부호없는 정수변환
	BOOL(a) / toBOOL(b)	0, 1부울 변환
	FLOAT(a) / toFLOAT(b)	32비트 실수형변환
	DOUBLE(a) / toDOUBLE(b)	64비트 실수형변환(v스크립트자료형과 동일함으로 의미없음)
보다 세밀한 자료형변환		
	INT8(a) / toINT8(b)	함수명의 숫자는 비트수를 나타냅니다.
	UINT8(a) / toUINT8(b)	
	INT16(a) / toINT16(b)	
	UINT16(a) / toUINT16(b)	
	INT32(a) / toINT32(b)	
	UINTt32(a) / toUINT32(b)	
	INT64(a) / toINT64(b)	
	UINT64(a) / toUINT64(b)	
	FLOAT32(a) / toFLOAT32(b)	
	FLOAT64(a) / toFLOAT64(b)	

V스크립트언어의 기본 데이터형(double)이 아닌 정보를 시스템메모리에서 읽고 쓰는 동작을 위해 아래 두가지 방법을 제안할 수 있습니다.
(1) 시스템 메모리 액세스 함수 WINC_PUT_DATA_.../WINC_GET_DATA_...함수를 사용하여 할 수 자료형 변환함수 사용.
(2) 자료변환함수 사용
※ 그러나 시스템메모리에서 문자열을 읽거나 쓸 경우 반드시 WINC_PUT_DATA_STR()/WINC_GET_DATA_STR() 함수를 사용해야 합니다.

2.13. v스크립트 인터프리터 관리 함수들

v스크립트 함수에는 아래와 같은 v스크립트실행과 관련된 몇가지 주요한 관리함수들을 제공합니다.

종류	함수명	설명
	libpath(path)	call() 및 include()함수에 대한 라이브러리 경로를 설정합니다.
	get_libpath()	현재 라이브러리 경로명을 얻습니다.
	get_defaultlibpath()	디폴트로 설정된 라이브러리 경로명을 얻습니다.
	get_programpath()	현재 프로그램이 실행중인 경로명을 얻습니다.
	include(codefile)	codefile 소스를 포함시켜 실행합니다.
	call(codefile, ...)	codefile 소스를 호출하여 실행합니다.
	compile(codestr)	codestr 문자열을 해석실행합니다.
	return(n)	현재 코드해석을 종료시킵니다.
	at(v)	v로 지정된 변수명(문자열)의 시스템메모리 위치(인덱스)값을 반환합니다. (구 get_varindex(v)함수와 동일)
	reset_arg()	call 호출에 사용한 인자메모리(스택)를 클리어합니다.
	wdt(t)	※r13.337 버전이후> 스크립트 실행을 감시합니다.. 주로 while문 등의 실행시간을 감시하며 t로 지정된 시간이 경과되면 while문에서 빠져나옵니다. 단, t가 음수값으로 지정되면 지정시간 초과시 스크립트전체를 종료시키고 메시지를 발생합니다. 0을 설정하면 wdt(워치독)기능을 비활성화시키며, 현재 wdt발생상태를 알린다.(음수이면 wdt발생) 리턴값은 이전에 설정된 워치독타임t값입니다.
	reset()	인터프리터 리셋
	alarm_reset()	인터프린터 알람리셋
	clr_svar(varname)	varname으로 선언된 정적변수를 삭제. varname이 “”로 전달되면 전체 정적변수를 삭제.

2.14. 메모리 관련

배열 등의 메모리 운용과 파일로딩 메모리 포인터 관련 함수들입니다.

종류	함수명	설명
	copy_array(d,s,n)	배열 전송 d = destination array s = source array n = array size
	set_array(d,n,v)	배열 설정 d = destination array n = array size v = set value
	shift_block(n,e,b,m,r)	시스템메모리블록을 시프트 및 로테이트합니다. n : 데이터블록 이동방식 n=0) 메모리블록 상향 시프트 n=1) 메모리블록 하향 시프트 e ; 블록 크기 b ; 블록 수 m ; 블록 배열메모리 r ; 인입, 인출 단위블록 메모리
	load_file_alloc(fname)	fname 의 파일을 로딩하여 메모리 할당한 포인터리턴. 3d 그래프 드로잉에서 stl파일등을 등록할때 사용됨.

2.15. 시간관련

시간지연과 시간, 날짜 관련 함수입니다.

종류	기호	설명
	delay(t)	t는 초단위. 시간을 지연합니다.
	wait(t)	t는 초단위.내부적으로 메세지를 처리하면서 대기합니다.
	getmsec()	1/1000초 단위로 프로그램이 시작된 이후의 msec시간을 얻습니다.
	getyear()	연도를 얻습니다.
	getmonth()	달수를 얻습니다.
	getday()	월 단위 일수를 얻습니다.
	getweekday()	주일단위 일수를 얻습니다.
	getyearday()	연단위 일수를 얻습니다.
	gethour()	일단위 시간을 얻습니다.
	getminute()	시간단위 분을 얻습니다.
	getsecond()	분단위 초를 얻습니다.

2.16. 사운드 관련

종류	함수	설명
	sound(soundfile)	soundfile은 경로명이 포함된 wav파일명 입니다.
	volume(v)	마스터볼륨으로써 0 ~ 100(%)를 설정합니다.
	beep(hz,t)	비프음(주파수,시간)

2.17. 공통 다이얼로그

종류	기호	설명
	file_dialog(fmt,m)	파일열기 다이얼로그입니다. fmt는 열기파일의 확장자형식(“*.nc")이고 m은 다이얼로그 생성방식입니다. 0이면 쓰기모드, 1이면 읽기모드, 2이면 현재폴더를 옮기지 않는 쓰기모드, 3이면 현재폴더를 옮기지 않는 읽기모드입니다.
	color_dialog(m,c)	색상선택 다이얼로그입니다. m은 다이얼로그 생성방식입니다. 0이면 디폴트칼라 c가 유효합니다. 1은 디폴트칼라를 무시합니다.
	keypad_dialog(s,m)	터치 입력용 다이얼로그입니다. s는 초기문자열이며 m은 다이얼로그 종류입니다. m = 1 : 숫자키패드, m=10 : 문자키패드
	input_dialog(m,s)	키보드 입력용 다이얼로그입니다. m은 다이얼로그 종류, s는 셋업문자열입니다. m=0; (다이어롤그 타입 디폴트) <셋업문자열포맷 : 셋업키문자열 = 셋업데이터, ... 반복> "password = 1, " // 패스워드입력으로 설정 “initstr = 123, " // 초기에 123 문자열을 디스플레이함. “title = Height, " // 입력창에 Height 문자표기 “titlecolor =RGB_RED, " // 타이틀칼라지정 “titlebkcolor = RGB_WHITE, " // 타이틀배경칼라지정 “editcolor = REG_BLACK, " // 편집폰트칼라지정 “editbkcolor = RGB_WHITE, " // 편집배경칼라지정 “datatype = 0" // 리턴데이터형, 0=수치값, 1=문자열

2.18. 메시지 및 로그기록

종류	기호	설명
	messagebox(n,p,t1,t2)	n=0~6 : 메시지박스종류 0=CNC상태메시지, 1=OK메시지, 2=OK/CANCEL, 3=ABORT/RETRY/IGNORE, 4=YES/NO, 5=YES/NO/CANCEL/ 6=RETRY/CANCEL p=0~4 : 표시아이콘타입 0=알림, 1=경고, 2=정보, 3=질의, 4=오류, (단, CNC상태메세지일 경우 0=알림, 1=주의, 2=알람, 3=메시지삭제) t1=창타이틀, t2=메시지 반환값 : 0=NO, 1=OK/YES, 2=CANCEL, 3=RETRY, 4=IGNORE, -1=ABORT,
	logfile(title,context)	로그파일에 title문자열과 context문자열의 로그를 기록합니다.

2.19. 시스템 명령

종류	기호	설명
시스템 외부실행	system(cmd)	cmd 문자열의 시스템명령을 실행합니다.
핫키 등록	register_hotkey(id,mod,key,str)	id = 0~199 (핫키등록번호) mod = 0x01(alt키), 0x02(ctrl키), 0x04(shift키) key = 숫자영문 대문자키코드(visual c++의 VK_ 코드) str = 핫키등록스트링

2.20. 그래프 관련함수

종류	기호	설명
	RGB(r,g,b)	r,g,b 는 각각 1바이트 red, green, blue 16진 칼라값입니다. (바이트순서 b,g,r 입니다.)
	RGB_R(c)	c값에서 RED값을 리턴합니다. 0~255
	RGB_G(c)	c값에서 GREEN값을 리턴합니다. 0~255
	RGB_B(c)	c값에서 BLUE값을 리턴합니다. 0~255
	RGB_SUM(c1, c2)	c1+ c2 칼라값을 리턴합니다. (R,G,B별로 연산)
	RGB_SUB(c1, c2)	c1- c2 칼라값을 리턴합니다. (R,G,B별로 연산)
	RGB_PROP(c, p)	c칼라의 세기를 p퍼센트값으로 조절합니다.
	rgb(c) rgb_r(c) rgb_g(c) rgb_b(c)	-c는 4바이트 16진 칼라값 입니다. (b,g,r바이트 순서를 r,g,b로 바꾸어줍니다.) rgb_r/g/b(c) 는 각각 red, green, blue값을 리턴합니다.
	setup_3d(g,c,in[],out[])	-g는 그래프 그룹 0~39 총 40개 그룹 -c는 명령문자열이며 아래의 명령등을 사용한다. init, on, off, mode, buff, clrbuff, link, arc_ang, arg_div, offset, offseti, translate, translatei, rotate, rotatei, rotate2, rotate2i, rotate3, rotate3i, scale, scalei, get_onscreen, get_mode, get_buffindex, get_link, get_arc_ang, get_arc_div, get_offset, get_move, get_rotate, get_rotate2, get_rotate3, get_scale, get_cur_model, get_last_model -in[] 은 전달인자, out[]은 결과값 ※자세한 내용은 별도의 3d 드로잉편을 참조합니다.
	draw_3d(g,c,in[],out[])	-g는 그래프 그룹 0~39 총 40개 그룹 -c는 명령문자열이며 아래의 명령등을 사용한다. start, end, clear, color, colori, loadidentity, pushmatrix, popmatrix, loadmatrix, multiplematrix, translate, rotate, scale, line, linestrip, triangle, trianglestrip, trianglefan, arc, solidarc, linei, linestripi, trianglei, trianglestripi, trianglefani, arci, solidarci, cone, solidecone, cube, solidecube, sphere, solidsphere, torus, solidetorus, text, texti, loadstl, moveto, lineto, arccw, arcccw, movetoi, linetoi, arccwi, arcccwi -in[] 은 전달인자, out[]은 결과값 ※자세한 내용은 별도의 3d 드로잉편을 참조합니다.
CUSTOM 작화관련 함수	find_ctrlno(w,name)	-w=커스텀작화윈도우번호, -name = 작화컨트롤 사용자 지정이름. 버튼, 에디터 등의 작화컨트롤에 지정된 이름(name=xxxx)과 동일한 컨트롤의 번호를 리턴합니다.

2.21. 그래프 매트릭스 수학관련

3그래픽 드로잉을 위한 간단한 동차행렬[4]x[4]이나 벡터[4]를 초기화하거나 연산하는 함수들입니다.

종류	함수명	설명
벡터연산	set_hvect(v,x,yz);	var v[4]; v=(x,y,z,1); 와 문법과 동일
	copy_hvect(v1,v2);	var v1[4], v2[4]; v1 = v2; 와 문법과 동일
	sum_hvect(v1,v2,v3);	v1[4] = v2[4] + v3[4]
	sub_hvect(v1,v2,v3);	v1[4] = v2[4] - v3[4]
	scale_hvect(v1, s, v2);	v1[4] = s * v2[4]
	rotate_hvect(v1, x,y,z, ang, v2)	v2벡터를 x,y,z벡터를 기준으로 ang(각도) 회전한 벡터를 v1[4]벡터에 저장한다.
	cross_hvect(v1,v2,v3);	v1[4] = v2[4] x v3[4]
	dot_hvect(v1,v2);	v1[4] = v2[4] dot v3[4]
	ang_hvect(v1,v2);	v1벡터와 v2벡터간의 사잇각(내적)
	normal_hvect(v1,v2);	v2를 normalize하여 v1에 저장
	norm_hvect(v1);	v1의 norm값을 얻는다.
	trans_hvect(v1, mat, v2);	v2[4]벡터를 mat[4][4]로 변환하여 v1[4]벡터에 저장한다.
동차행렬연산	ident_hmat(mat);	mat를 identifier 한다.
	copy_hmat(mat1, mat2);	mat2를 mat1에 복사
	transpose_hmat(mat1, mat2);	mat2을 transpose하여 mat1에 저장
	translate_hmat(mat1, x,y,z);	mat1의 이동변환
	scale_hmat(mat, x,y,z);	mat1의 스케일변환
	rotate_hmat(mat, x,y,z, ang);	mat1의 회전변환
	trans_hmat(mat1, mat2, mat3)	mat1 = mat2 x mat3

	solve_mat(n, mat, retv);	n은 근의 수, mat는 연립방정식행렬, ret는 결과벡터
		
		예> 3원 1차연립방정식이 아래와 같을 때
		a1x + b1y + c1z = d1;
		a2x + b2y + c2z = d2;
		a3x + b3y + c3z = d3; 일 때 (단, a1, b1...는 상수)
		
		// 변수선언
		var ret, n(3);
		var mat[3][4], retv[3];
		// 연립방정식행렬 초기화
		mat[0] = (a1, b1, c1, d1);
		mat[1] = (a2, b2, c2, d2);
		mat[2] = (a3, b3, c3, d3);
		// 계산 -> 결과값(해)는 retv에 저장
		//만약 ret값이 -1 이면 해를 구할 수 없음.
		ret = solve_mat(n, mat, retv);

2.22. WINDOWS 표준 DLL 연결

윈도우즈 정규DLL함수를 호출합니다.

- DLL 함수 프로토타입> double DLL_Func(void *pSysData[], double *inp, double *outp);

- pSysData[]는 포인터배열로써
  pSysData[0]은 시스템메모리맵 포인터입니다.
  pSysData[1]은 V스크립트 인터프리팅 함수1: double Run_VScirpt(char *codestr);
  pSysData[2]는 V스크립트 인터프리팅 함수2: double Run_VScirptW(TCHAR *codestr);
  그 외 pSysData[] 포인터는 예약되어 있어 사용자가 임의적으로 사용해서는 안 됩니다,

- 시스템 메모리맵은 배정도실수의 포인터배열 (double *sysMemoryMap[]) 입니다.
  시스템 메모리맵 인덱스범위는 0~99,999 입니다.

- inp, outp 사용자가 사용할 있는 임의 전달 파라메터 입니다.

종류	함수명	설명
	load_dll(dllfile)	dll파일 로딩, 핸들값을 리턴합니다. 0 이면 오류 일반적으로 파워온 초기화 단계나 리셋단계에서 실행되는 것이 바람직합니다.
	exe_dll(h,func,p1,p2)	dll함수 실행함수 h = dll 핸들 func = 실행함수명 p1,p2 = 사용자 임의 입력 파라메터
	free_dll(h)	dll파일 해제 모션마스터 종료단계에서 실행되는 것이 바람직합니다.

자세한 dll 연결 내용은 별도의 예제와 설명서를 홈페이지 자료실에서 참조하실 수 있습니다.

-끝-