CNRLAB

CIM&ROBOTICS LABORATORY

2주차 ROBOTC 기초 프로그래밍

2015.09.07

기초로봇공학실험

NXT Brick

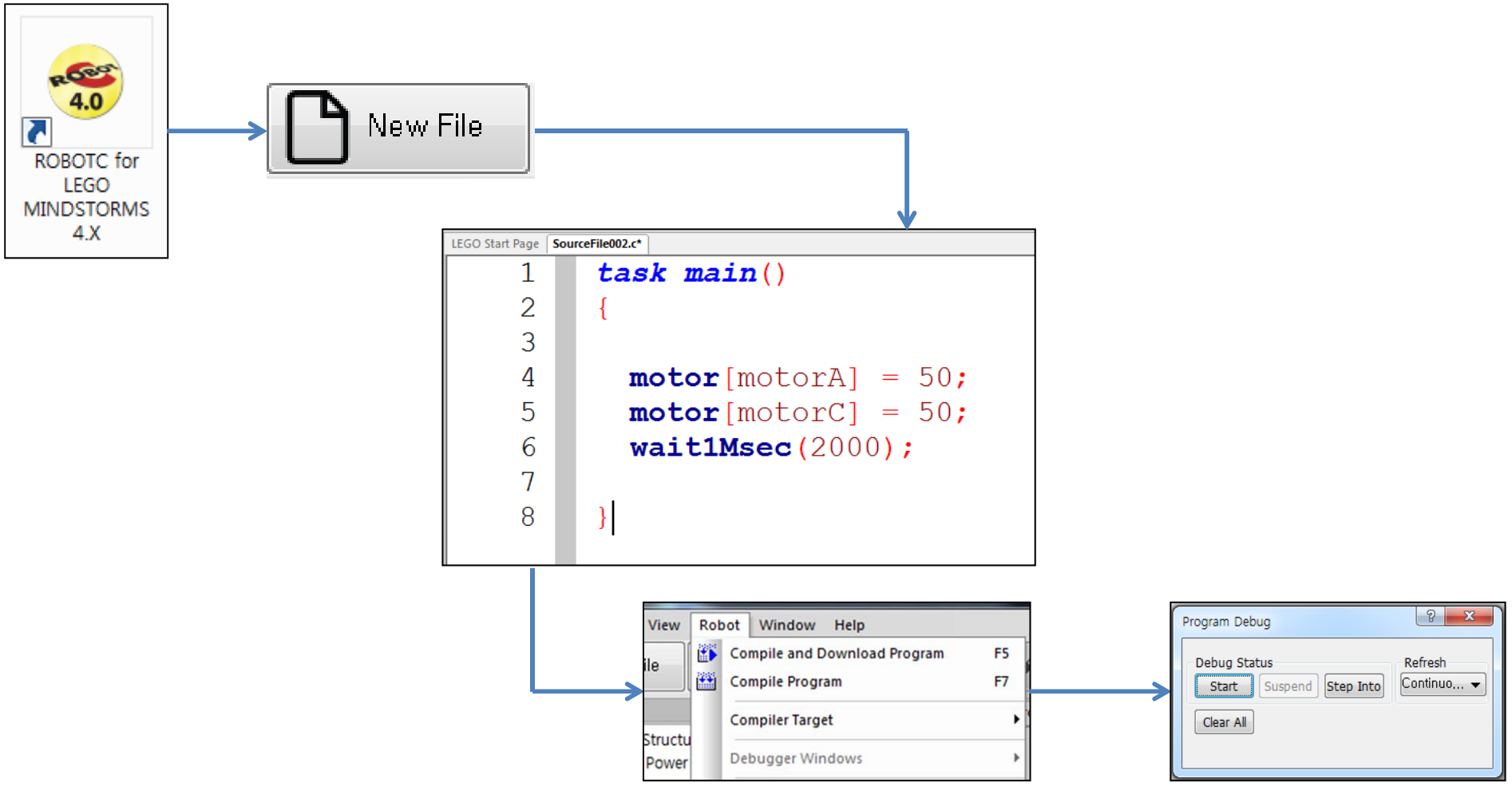
꺼진 상태 : 전원 켜기
켜진 상태 : 가운데 메뉴 실행

좌측의 메뉴로 넘어감

우측의 메뉴로 넘어감

취소 / 최상위 메뉴에서 끄기

프로그램 사용법


ROBOTC for LEGO MINDSTORMS 4.X
New File
LEGO Start Page
SourceFile002.c*
1 task main()
2 {
3
4 motor[motorA] = 50;
5 motor[motorC] = 50;
6 wait1Msec(2000);
7
8 }
View
Robot
Window
Help
Compile and Download Program F5
Compile Program F7
Compiler Target
Debugger Windows
Program Debug
Debug Status
Refresh
Start
Suspend
Step Into
Continuo...
Clear All

ROBOTC 기본메뉴 및 단축키(참고)

File

- New(Ctrl + N) : 새로운 문서작성
- Open and Compile : 불러오기와 컴파일을 동시에 실행
- Open Sample Program : 샘플 프로그램 불러오기
- Save(Ctrl+S) : 작성된 소스를 RobotC 파일로 저장
- Save As : 다른 이름으로 저장
- Print(Ctrl+P) : 작성된 프로그램을 인쇄
- Print Preview : 인쇄할 프로그램 미리 보기
- Page Setup : 현재 작성된 프로그램 출력페이지 설정

Edit

- Undo (Ctrl+Z) : 실행취소, 되돌리기 기능
- Redo (Ctrl+Y) : 실행취소된 것을 되돌리기 기능
- Cut (Ctrl+X) : 잘라내기
- Copy (Ctrl+C) : 복사하기
- Paste (Ctrl+V) : 붙여넣기
- Find (Ctrl+F) : 단어나 문장 찾기
- Find Next (F3) : 다음 위치의 단어나 문장 찾기
- Find and Replace (Ctrl+H) : 찾아 바꾸기
- Advanced : 들여쓰기 기능을 활성화시키거나 취소하고, 기본양식을 설정

View

- Font Increase (Ctrl +‘ +’) : 폰트사이즈 증가
- Font Decrease (Ctrl +‘ -’) : 폰트사이즈 감소
- References : RobotC 프로그램의 환경을 설정하는 곳으로 플래폼의 설정, 전원과 센서의 사용, 프로그램 설치경로, 파일의 저장경로, 글꼴 및 색상 설정, 버그설정 등 프로그램을 위한 환경을 설정

ROBOTC 기본 규칙

- ROBOTC는 표준 **C언어 프로그램 규칙**을 따르는 text 기반의 프로그램 언어.
- 파랑색, 보라색 글씨 : ROBOTC가 그 단어를 매우 중요한 단어로 인식한다는 뜻
- 검정색 글씨 : 잘못 작성되었을 가능성이 매우 높음
- ROBOTC가 인식하는 키워드(keyword)는 자동으로 색상이 결정되어 표시.
- ROBOTC 컴파일러는 **대문자와 소문자를 엄격하게 구분**.
- 프로그램의 실행순서는 **1번 줄부터 순차적으로 실행**.
- 스페이스(space), 탭(tab)은 프로그램의 가독성을 높이기 위해서 적절한 사용. (프로그램 영향X)
- **세미콜론(;)**은 모든 ROBOTC 프로그램 문장의 끝.

ROBOTC 변수선언 유의사항

변수 선언 시 유의 사항

- 대문자(A~Z), 소문자(a~z), 숫자(0~9), 밑줄문자' _ '를 사용한다.
- 변수명은 숫자로 시작할 수 없다.
- 변수명은 기본 예약어 또는 함수명을 사용할 수 없다.
- 변수명은 대·소문자를 다르게 구분한다.
- 변수명은 한글로 사용할 수 없다.

ROBOTC 자료형 – Data type

데이터형	데이터형	메모리 크기	표현 범위	표현 범위
문자형 (character)	char	1바이트	-128 ~ 127	양수, 0, 음수
	unsigned char	1바이트	0 ~ 255	0, 양수
정수형 (integral)	int	2바이트	-32,768 ~ 32,767	양수, 0, 음수
	unsigned int	2바이트	0 ~ 65535	0, 양수
	long	4바이트	-2,147,483,648~ 2,147,483,647	양수, 0, 음수
	unsigned long	4바이트	0 ~ 4294967295	0, 양수
실수형 (floating)	float	4바이트	1.2E-38 ~ 3.4E38	양수, 0, 음수

- 프로그램에서 처리하고자 하는 자료의 형태를 의미
- NXT 메모리 영역의 크기 결정

ROBOTC 자료형 – Data type

화면 출력 서식

서식	자료형	의미
%d	Int	정수형 자료
%u	unsigned int	부호 없는 정수형 자료
%c	Char	문자형 자료
%s	String	문자형 자료
%f	float	실수형 자료

Coding

```
nxtDisplayTextLine(1,"x is %d", x);
```

1번 Line　　출력 문구　　정수형 변수 x값 출력

ROBOTC 자료형 – Data type

정수형 변수

```
task main()
{
  int x, y;

  x = 100;

  y = x*3;

  nxtDisplayTextLine(1,"x is %d",x);
  nxtDisplayTextLine(2,"y is %d",y);
  wait1Msec(3000);

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

ROBOTC 자료형 – Data type

실수형 변수

```
task main()
{
  float x, y;

  x = 1234.567;

  y = x*3;

  nxtDisplayTextLine(1,"y is %.2f",y);
  wait1Msec(3000);

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

ROBOTC 자료형 – Data type

문자형 변수

```
task main()
{
  char x;

  x = 65;

  nxtDisplayTextLine(1,"A is %d",'A');
  nxtDisplayTextLine(2,"x is %c", x);
  wait1Msec(3000);

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

ASCII ?

ROBOTC 자료형 – Global variable

❖ **전역변수**

- 함수의 밖에 정의된 변수
- 프로그램 내 모든 함수들이 사용 가능한 변수
- 함수들 간에 공유 데이터가 필요할 경우 사용

```
int x, y;
int add()
{
  return x+y;
}

task main()
{
  x = 3;

  nxtDisplayTextLine(1,"x is %.2d",x);
  nxtDisplayTextLine(2,"x+y is %d",add());
  wait1Msec(3000);
}
```

ROBOTC 자료형 – 연산자

• 보다 작다 〈	ex) A 〈 B	의미: A가 B보다 작다
• 보다 크다 〉	ex) A 〉 B	의미: A가 B보다 크다
• 작거나 같다 〈=	ex) A 〈= B	의미: A가 B보다 작거나 같다
• 크거나 같다 〉=	ex) A 〉= B	의미: A가 B보다 크거나 같다
• 같다 ==	ex) A == B	의미: A와 B는 같다.
• 다르다 !=	ex) A != B	의미: A는 B와 같지 않다.

```
task main()
{
  int A = 10, B=5;
  if (A!=B)
  {
    nxtDisplayTextLine(1,"True");
    wait1Msec(3000);
  }
}
```

ROBOTC 자료형 – 연산자

- AND ex) A && B // A와 B가 모두 참일 때 참
- OR ex) A || B // A와 B중 하나이상이 참일 때 참
- NOT ex) !A // A의 현재 상태의 반대

```
task main()
{
  int A = 10, B = 5;
  if (A > 10 && B < 10)
  {
    nxtDisplayTextLine(1,"True");
    wait1Msec(3000);
  }
}
```

프로그램 기초

➢ 예제 1-1 : 2초 동안 전진

```
task main()
{
  motor[motorA] = 75;
  motor[motorC] = 75;
  wait1Msec(2000);

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

- motor[motorA] = 75;

: 포트A의 모터를 75%의 파워로 정회전

모터출력 : -100% ~ 100%

- wait1Msec(2000);

: 2000ms만큼 이전 동작 유지

1ms = 10^{-3}sec

프로그램 기초

➢ 예제 1-2 : 2초 동안 전진, 2초 동안 후진

- ✓ 조건 : 후진할 때에는 NXT에 내장된 사운드 소리를 내면서 돌아올 것!

프로그램 기초

- 예제 1-3 : 2초 전진, 90도 회전, 2초 전진

프로그램 기초

➢ 예제 1-4 : LCD 화면 출력

✓ 조건 : NXT LCD화면에 “Hello World” 라는 문구 출력!

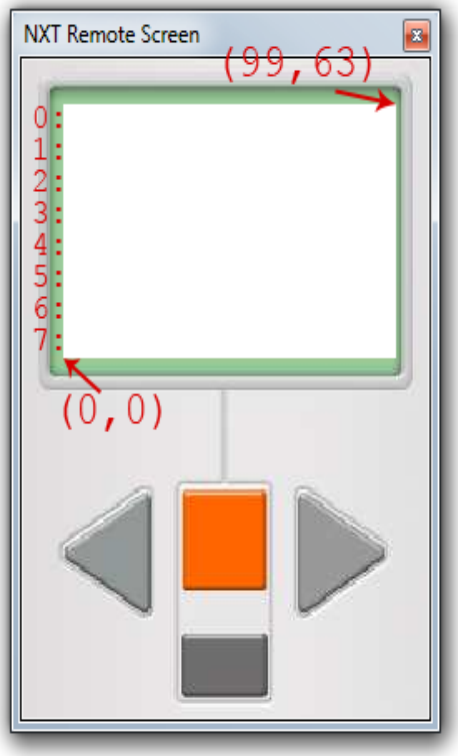
```
task main()
{
  nxtDisplayTextLine(4, " Hello World ");
  wait1Msec(5000);

  Playsound(soundBeepBeep);
  wait1Msec(500);
}
```

프로그램 기초

➢ 예제 1-5 : LCD 화면 출력(2)

✓ 조건 : NXT LCD화면에 “BIG : ABC” 라는 문구를 큰 폰트로 출력!

```
task main()
{
  nxtDisplayBigStringAt(0, 30, " Big : %s ", "ABC");
  wait1Msec(5000);

  Playsound(soundBeepBeep);
  wait1Msec(500);
}
```



프로그램 기초

➢ 예제 1-6 : 타이머를 활용한 제어

✓ 조건 : 타이머를 활용하여 5초 동안 직진!

```
task main()
{
  clearTimer(T1);

  while ( time1[T1] < 5000 )
  {
    motor[motorA] = 50;
    motor[motorC] = 50;
  }
}
```

- ClearTimer(T1);

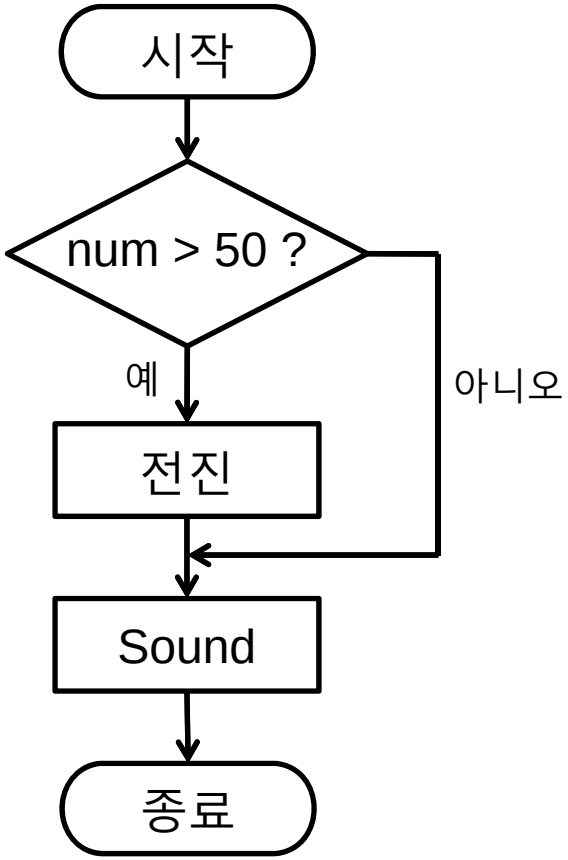
: 타이머T1을 0으로 초기화

네 개의 타이머 사용 가능 (T1 ~ T4)

time1[T1]으로 시간 사용 가능

ROBOTC 제어문

If 문

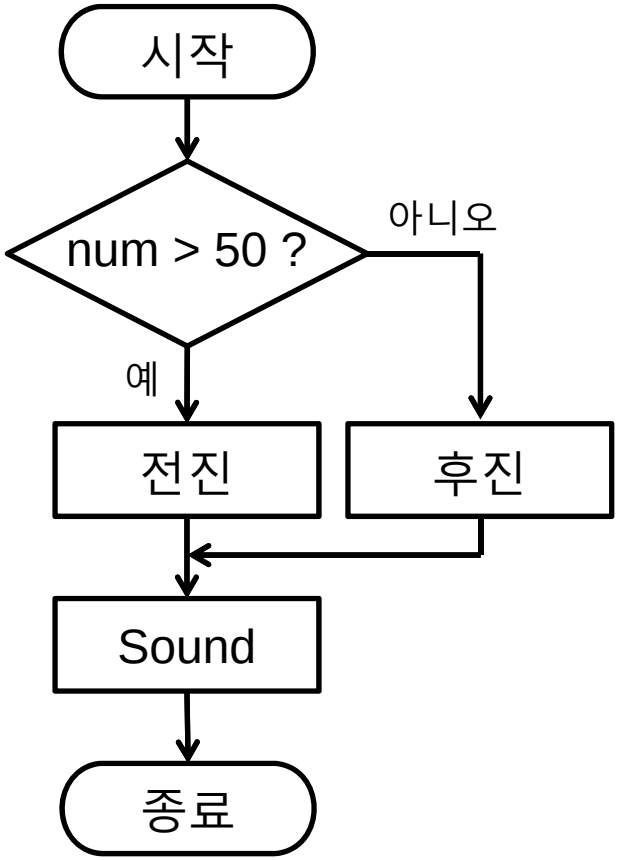



```
task main()
{
  int num;

  num = random(100);

  if( num > 50 )
  {
    motor[motorA] = 50;
    motor[motorC] = 50;
    wait1Msec(1000);
  }

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

```
if ( 조건식 )
{
      문장 ;
}
```

ROBOTC 제어문

If - else 문

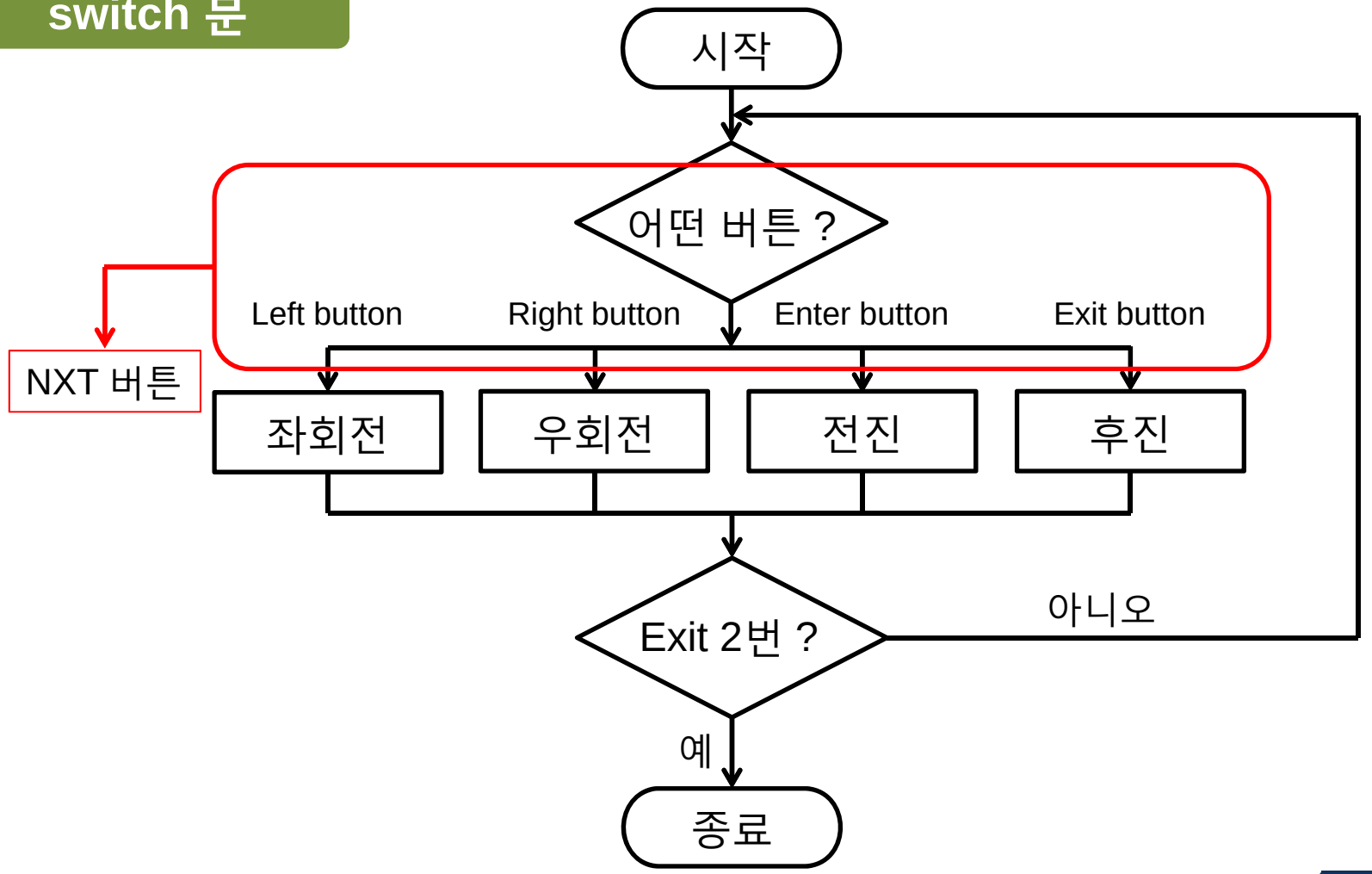



```
task main()
{
  int num;

  num = random(100);

  if( num > 50 )
  {
    motor[motorA] = 50;
    motor[motorC] = 50;
    wait1Msec(1000);
  }
  else
  {
    motor[motorA] = -50;
    motor[motorC] = -50;
    wait1Msec(1000);
  }

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

```
if ( 조건식 )
{
      문장 1 ;
}
else
{
      문장 2 ;
}
```

ROBOTC 제어문

switch 문



ROBOTC 제어문

switch 문

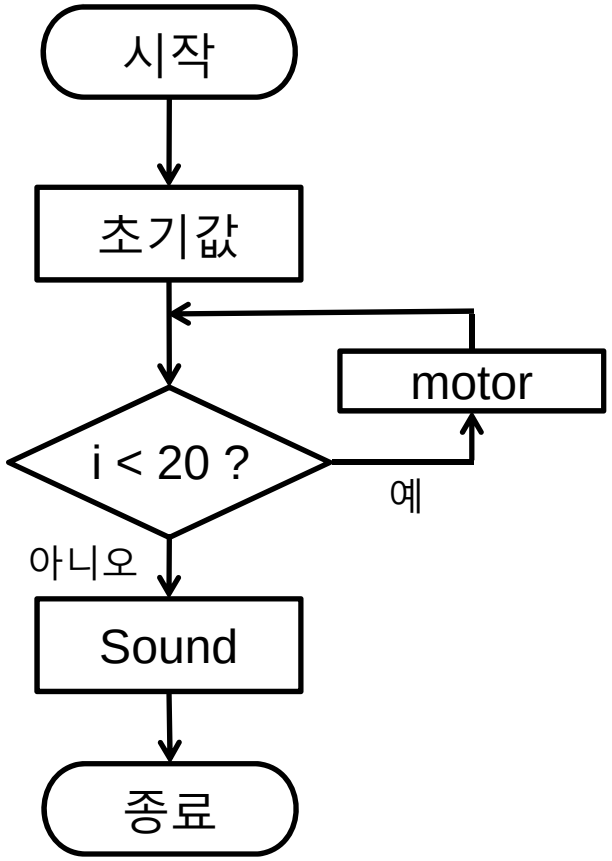
```
task main()
{
  nNxtExitClicks = 2;

  while(true)
  {
    switch(nNxtButtonPressed)
    {
      case kLeftButton :
       motor[motorA] = 0;
       motor[motorC] = 100;
       break;

      case kRightButton :
       motor[motorA] = 100;
       motor[motorC] = 0;
       break;

      case kEnterButton :
       motor[motorA] = 100;
       motor[motorC] = 100;
       break;

      case kExitButton :
       motor[motorA] = -100;
       motor[motorC] = -100;
       break;

      default :
       motor[motorA] = 0;
       motor[motorC] = 0;
    }
  }
}
```

```
switch ( 수식 )
{
      case 값1 :
            문장1 ;
      case 값2 :
            문장2 ;
            ⋮
      default :
            문장N ;
}
```

- nNxtExitClicks = 숫자 : '숫자' 만큼 취소 버튼 연속 클릭 시 프로그램 종료
- nNxtButtonPressed : NXT의 어떤 버튼이 눌렸는지 알려주는 변수

ROBOTC 제어문

for 문

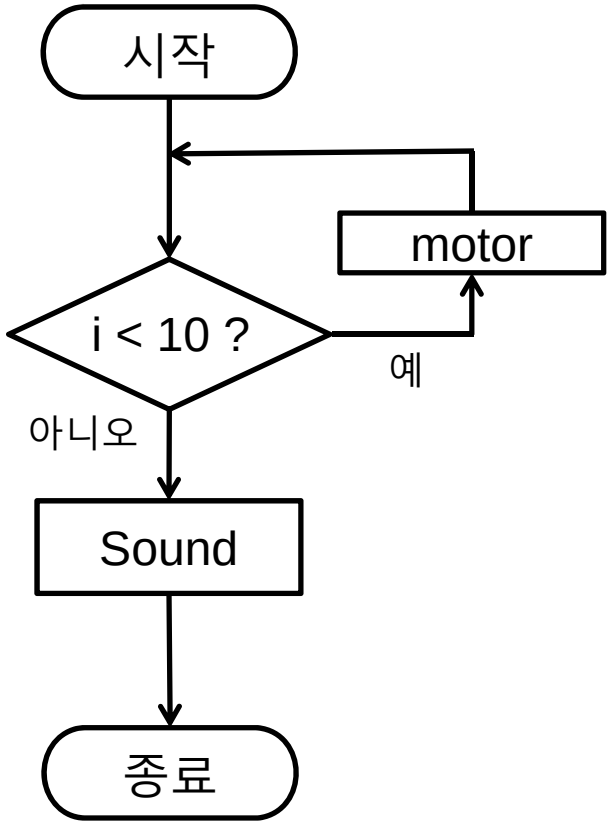



```
task main()
{
  int i;

  for(i=0 ; i<20 ; i++)
  {
    motor[motorA] = 75 ;
    motor[motorC] = 75 ;
    wait1Msec(500);
    motor[motorA] = -75 ;
    motor[motorC] = -75 ;
    wait1Msec(500);
  }

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

```
for (초기값 ; 조건식 ; 증감값)
{
      반복문장 ;
}
```

ROBOTC 제어문

while 문

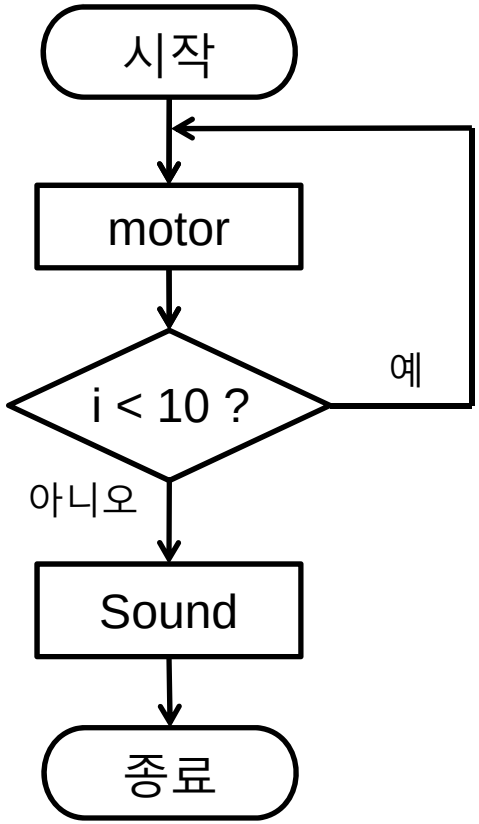



```
task main()
{
  int i;

  while( i<10 )
  {
    motor[motorA] = 75 ;
    motor[motorC] = 75 ;
    wait1Msec(500);
    motor[motorA] = -75 ;
    motor[motorC] = -75 ;
    wait1Msec(500);
    i++ ;
  }

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

```
while ( 조건식 )
{
      반복문장 ;
}
```

ROBOTC 제어문

do – while 문




```
task main()
{
  int i = 0;

  do
  {
    motor[motorA] = 75 ;
    motor[motorC] = 75 ;
    wait1Msec(500);
    motor[motorA] = -75 ;
    motor[motorC] = -75 ;
    wait1Msec(500);
    i++ ;
  }
  while( i<10 );

  PlaySound(soundBeepBeep);
  wait1Msec(500);
}
```

```
do
{
      반복문장 ;
}
while ( 조건식 ) ;
```

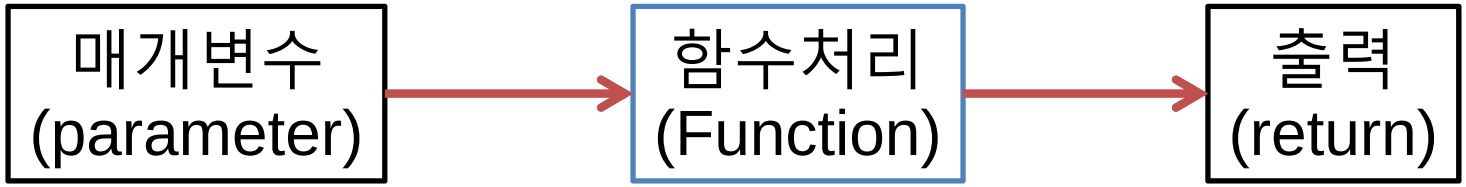
ROBOTC 제어문

- Questions !
  - if 문 vs switch 문
  - for 문 vs while 문

함수(Function)

➢ **함수**란 ?

: 프로그램 내에서 반복적으로 많이 사용되는 작업을 작은 모듈로 만드는 것



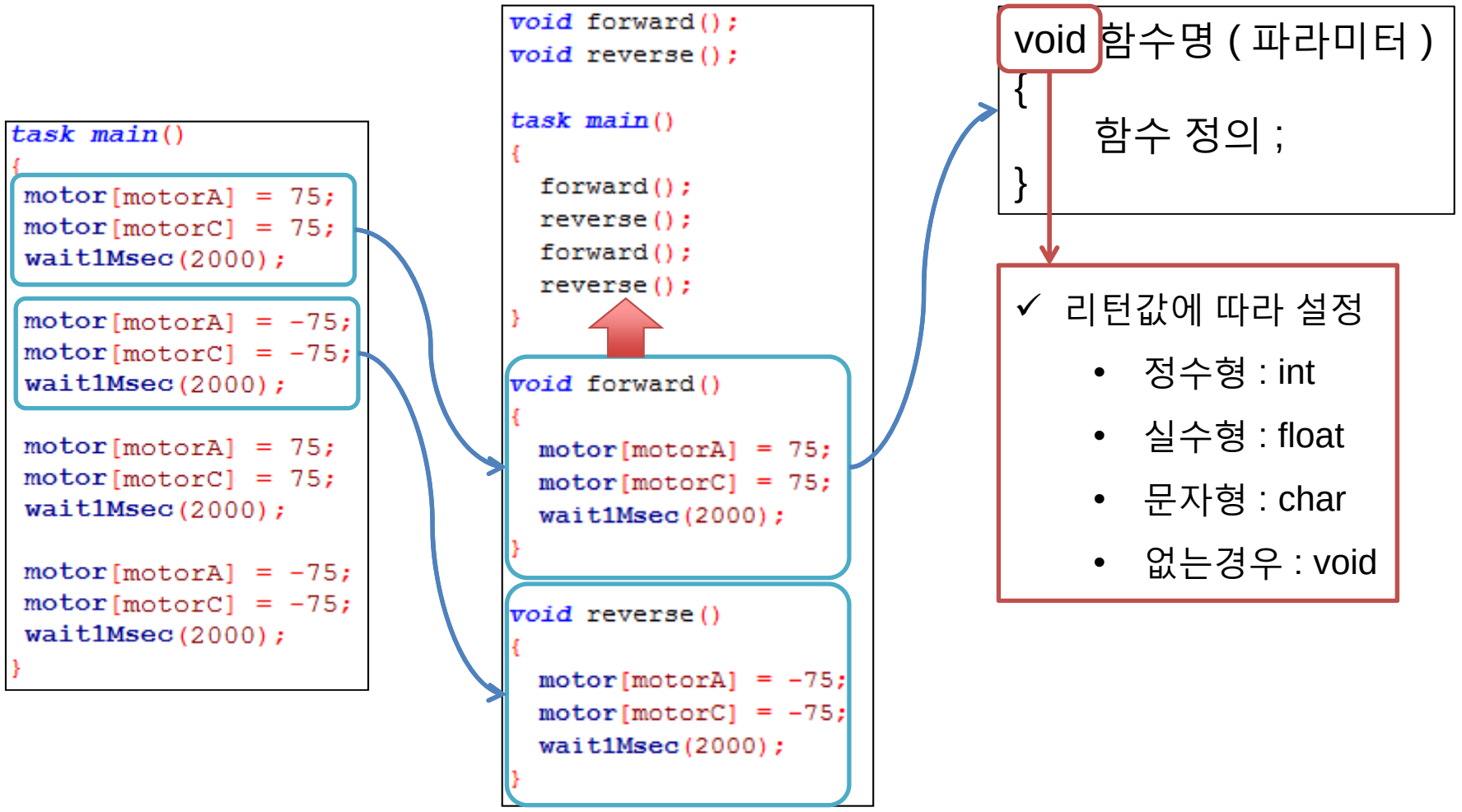
✓ 장점

- 작은 단위로 모듈화 가능
- 간편한 유지보수
- 재사용율 증가

✓ 종류

- main 함수
- 표준 라이브러리 함수 (제작사에서 제공)
- 사용자 정의 함수

함수(Function)


task main()
{
motor[motorA] = 75;
motor[motorC] = 75;
wait1Msec(2000);
motor[motorA] = -75;
motor[motorC] = -75;
wait1Msec(2000);
motor[motorA] = 75;
motor[motorC] = 75;
wait1Msec(2000);
motor[motorA] = -75;
motor[motorC] = -75;
wait1Msec(2000);
}
void forward();
void reverse();
task main()
{
forward();
reverse();
forward();
reverse();
}
void forward()
{
motor[motorA] = 75;
motor[motorC] = 75;
wait1Msec(2000);
}
void reverse()
{
motor[motorA] = -75;
motor[motorC] = -75;
wait1Msec(2000);
}
void 함수명 (파라미터)
{
함수 정의 ;
}
✓ 리턴값에 따라 설정
• 정수형 : int
• 실수형 : float
• 문자형 : char
• 없는경우 : void

함수(Function)

➢ 예제 2-1 : 매개변수 입력을 통한 함수 구현

✓ 조건 : 모터출력(speed)과 지속시간(time)을 매개변수로 하는 모터구동 함수 생성 !
리턴값은 없으므로 void !

함수(Function)

➢ 예제 2-2 : 매개변수 입력을 통한 함수 구현(2)

✓ 조건 : 정수형 리턴값을 갖는 함수 생성
두 개의 파라미터(정수형)의 합을 LCD에 3초간 출력

함수(Function)

#include 문

include " 파일명 "

" 파일명 " 내의 프로그램

```
#include "test2.c"

task main()
{
  forward();
  reverse();
}
```

```
void forward()
{
  motor[motorA] = 50;
  motor[motorC] = 50;
  wait1Msec(2000);
}

void reverse()
{
  motor[motorA] = -50;
  motor[motorC] = -50;
  wait1Msec(2000);
}
```

함수(Function)

매크로 = Replace ! , Not change !

```
#define SPEED 50
#define TIME 2000

task main()
{
  motor[motorA] = SPEED;
  motor[motorC] = SPEED;
  wait1Msec(TIME);
}
```

#define 상수이름 값

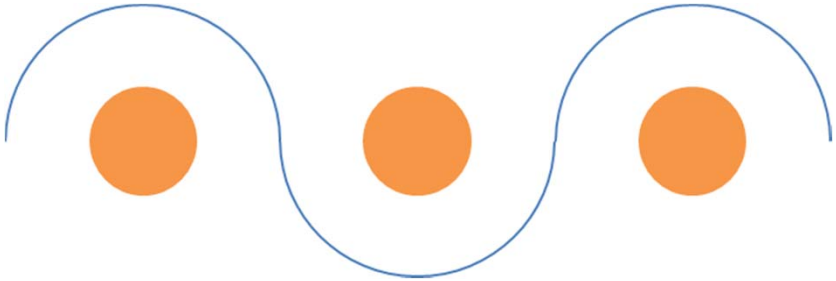
#define 함수이름(파라미터) 함수정의;

줄 바꿈 시, \ 를 입력

```
#define forward(speed,time)motor[motorA]=speed;\
motor[motorC]=speed;wait1Msec(time);

task main()
{
  forward(50,2000);
}
```

< 2주차 미션 >

1. NXT 엔터버튼 클릭 수 만큼 전진, 후진 반복
   - 속도 및 지속시간은 임의로 선택
   - 엔터버튼 클릭 후 5초간 대기 시 작동
2. NXT 좌우 버튼으로 속도 조절
   - 속도 및 지속시간은 임의로 선택
   - 좌우 버튼 모두 세 번 이상 push 일 경우 5초간 정지
3. 3개의 컵 통과
   - 취소버튼 1번 클릭 시 수행 (2번 클릭 시 Exit)



< 2주차 미션 >

➢ Performance

항목	세부 내용	배점
1번 수행	클릭 수와 반복 수의 일치	1
	전진, 후진 작동 여부	1
	5초 대기 후의 작동 여부	1
2번 수행	감속 및 가속 여부	1 X 2
3번 수행	컵 통과 개수	1 X 3
	취소버튼 클릭에 따른 작동 여부	2

➢ Algorithm & Programming

항목	세부 내용	배점
프로그램 능숙도	if / switch / 반복구문 / 함수 / 매크로 → 사용 여부에 따른 평가	0.6 X 5
순서도	미션 수행을 위한 순서도	2
컵 통과	좌, 우 속도 결정의 수학적 접근	5

과제(3주차 제출)

금주 **일요일(9.13)**까지
hshhln5@gmail.com 에 제출

예비 레포트

✓ 터치센서

- 작동원리 / 터치센서종류 / 적용분야

결과 레포트

- 로봇 구동 알고리즘 설명
  ① Source Code
  ② 순서도
- Discussion
  ① 기술적 문제점
  ② 문제 해결 방안