KOROBO3 応用編　第１章

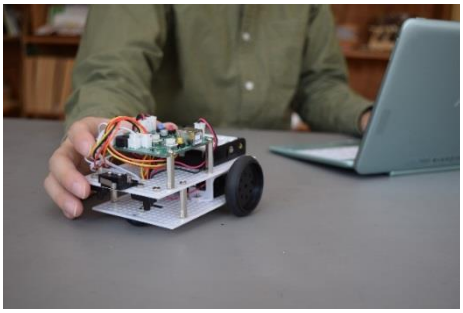
細かい制御をやってみよう

高度なプログラミングにチャレンジ！

基本編ではKOROBOを制御するためにプログラミングの基礎となる以下のことを学んできました。

- 順次実行：プログラムが行う処理は先頭から順に実行される
- 繰り返し：同じ動作を複数回、もしくは無限に行う
- 条件分岐：条件を満たしているかによって行動を変える

応用編ではKOROBOをもっと自由自在に動かすことができるよう、プログラミングのより深い部分を学んでいきます。付録の「アイコン一覧」「覚えると便利な操作テクニック」にはプログラミングに必要な情報が載っていますので、先に読んでおきましょう。

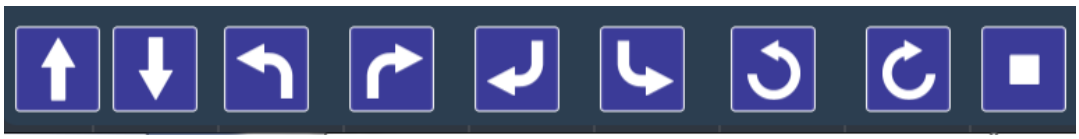


プログラムの新規作成

ロボットのプログラムを新しく作るときは、ポータルサイト上部に表示される「プログラミング環境 アドバンスト」のカードをタップ。「アドバンスト」は「ベーシック」よりも使えるアイコンがたくさん増えています。複雑なプログラムを組んで、より細かい制御を行うことができるようになります。

前進・後進・停止をしてみよう

「プログラミング環境 ベーシック」では、KOROBOを移動・回転させるときには下図のような矢印で進行方向や回転方向が示されているアイコンを使いました。



ベーシックで使われていた移動・回転のアイコン

「プログラミング環境 アドバンスト」では、KOROBOを移動・回転させるときには下図のような3種類のアイコンを使ってモーターを制御します。

モーター ON/OFF

モータースピード

モーターブレーキ

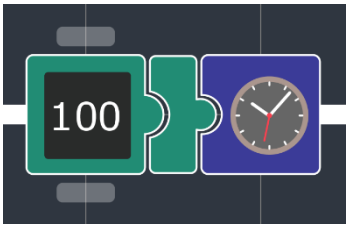
また、KOROBOを走らせる時間はベーシックのときと同じように時計マークの[待機]アイコンを使いますが、使い方はベーシックとアドバンストとで異なりますので注意してください。

待機　単位：0.1秒
（ベーシック）

待機　単位：ミリ秒
（アドバンスト）

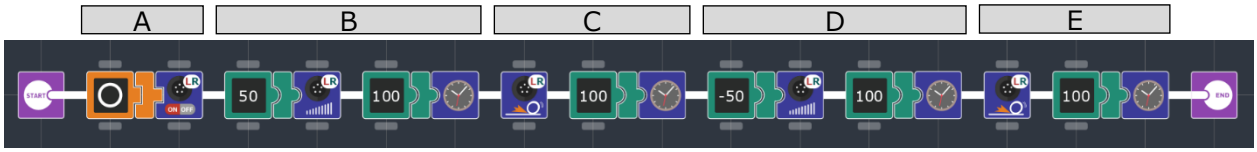
アドバンストではより細かい秒数を設定できるよう、ミリ秒単位で待機時間を設定できるようになります。「ミリ」というのは「1000分の1」という意味なので、例えば100という数値を[待機]アイコン

に入力すると、$\frac{1}{1000} \times 100 = 0.001 \times 100 = 0.1$ となり、0.1秒間待機することになります。



0.1秒間待機させる

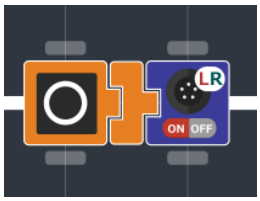
それでは、試しに「前進→停止→後進→停止」をそれぞれ0.1秒ずつ行うプログラム(例1)を作ってみます。
プログラム(例1)を作ったら動作テストをしてみましょう。



（例1）前進→停止→後進→停止 を0.1秒（100ミリ秒）ずつ行う

プログラムを部分ごとに細かく見てみましょう。

A では[モーター ON/OFF]アイコンに[○]を入力することで、左右のモーターの動作状態をONに設定します。モーターの動作状態の初期値は[×]（モーターOFF）となっているので、モーターを動かすときには必ず[○]を入力してモーターONとする必要があります。[モーター ON/OFF]アイコンのパラメータは初期値の[LR]のままにしましょう。



[A] 左右モーターON

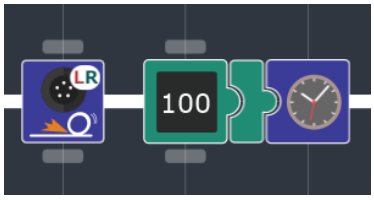
B では[モータースピード]アイコンに[数値定数(50)]を入力することでモーターが回転するスピードを設定しています。[待機]アイコンには[数値定数(100)]を入力することで0.1秒（100ミリ秒）待機させるプログラムを作っています。

[モータースピード]アイコンには-100～100の数値を入力することができ、1～100のときは正転（前進する方向）、-100～-1のときは逆転（後進する方向）にモーターが回転します。この数値の大きさはモーターの最大速度の何%で動かすかを意味しています。0のときはモーターの回転が止まります。

そのため、 B は「前進（スピード50%）を0.1秒間続ける」という動作をするプログラムとなります。

[B] 前進（スピード50%）を0.1秒間続ける

C では、[モーターブレーキ]アイコンによって左右のモーターにブレーキをかけて（スピードが0となり）止まります。[待機]アイコンに[数値定数(100)]を入力して0.1秒間止まるようにしています。 E も同様にブレーキをかけます。



[C] 停止状態を0.1秒間続ける

D では、[モータースピード]アイコンに[数値定数(-50)]を入力し、 [待機]アイコンに[数値定数(100)]を入力することで「後進（スピード50%）を0.1秒間続ける」という動作をします。 [モータースピード]の数値がマイナスなので後進（逆回転）します。

[D] 後進（スピード50%）を0.1秒間続ける

KOROBO3 応用編　第２章

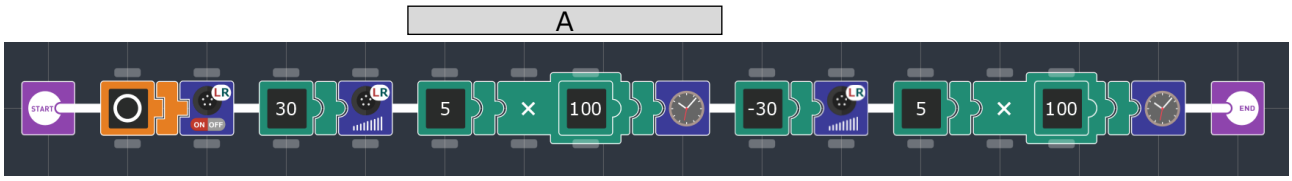
KOROBOに計算をさせてみよう

大きな数を使いたいときは…？

第1章ではKOROBOを前進・後進・停止させるプログラムを作りました。ここでは100ミリ秒(0.1秒)ずつ動作させましたが、実際にロボットを動かすときには0.5秒とか、1秒とか動くようにしたいことが多いです。
でも、このPalette IDEの数値定数のアイコンは-100～100の範囲しか値を設定することができないので、このままでは500ミリ秒(0.5秒)や1000ミリ秒(1秒)を設定することができません。
第2章では100より大きい数を使いたいときにどうすればいいかを学んでいきます。

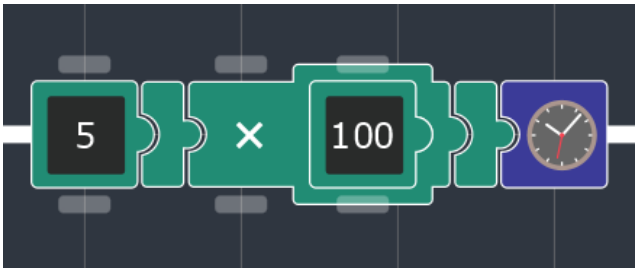
前進(0.5秒)→後進(0.5秒)をしてみよう

下図のような「0.5秒間前進→0.5秒間後進」するプログラムを作ります。



(例2) 前進(0.5秒)→後進(0.5秒)を行う

A を拡大して見てみましょう。



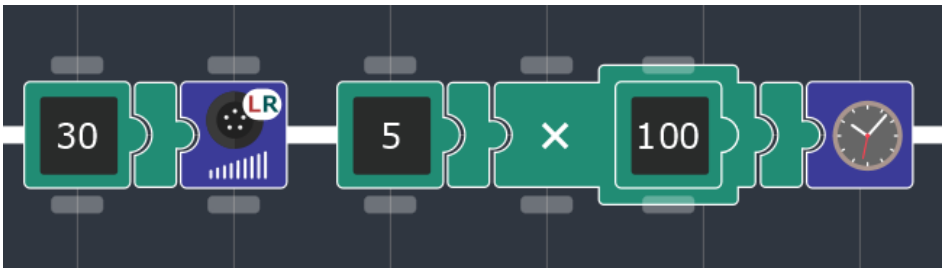
[A] 500ミリ秒(0.5秒)待機する

ここでは[算術演算]アイコンが使われています。[算術演算]アイコンはアイコンの左側に入力した数値と、内側に入力した数値を四則演算（足し算、引き算、掛け算、割り算）してその結果を出力します。パラメータボタンをクリックすると算術記号(＋,－,×,÷)を切り替えることができます。

算術演算アイコン

(例2)のプログラムは左側に[5]、内側に[100]を入力しており、算術記号が×(掛け算)となっているため、出力される値は$5 \times 100 = 500$となります。

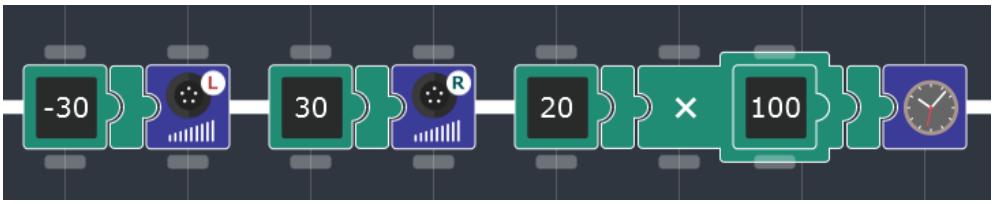
そのため、[待機]アイコンには[500]が入力され、 A の部分は「500ミリ秒(0.5秒)待機する」という意味になり、[モータースピード]アイコンも含めると、「スピード30%で0.5秒前進する」という動作をすることになります。その後の後進する部分についても同様です。



500ミリ秒(0.5秒)前進する

やってみよう

「左回転(2秒)→右回転(2秒)」の動きをするプログラムを作ってみましょう。ヒントとして、2秒間左回転するプログラムを下図に示します。回転させるために[モータースピード]のアイコンはLとRそれぞれ別の数値が入っています。



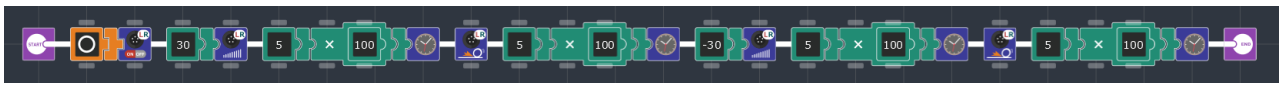
2000ミリ秒(2秒)左回転する

KOROBO3 応用編　第３章

数値を記憶させてみよう

待機時間を毎回計算させるのが大変なのですが…

第2章では掛け算を使うことで大きな数を扱えるようになりました。これでモーターを動かす時間を好きな値に設定することができますが、「前進→停止→後進→停止(それぞれ0.5秒ずつ)」のプログラムを作ってみると、下図のようになります。



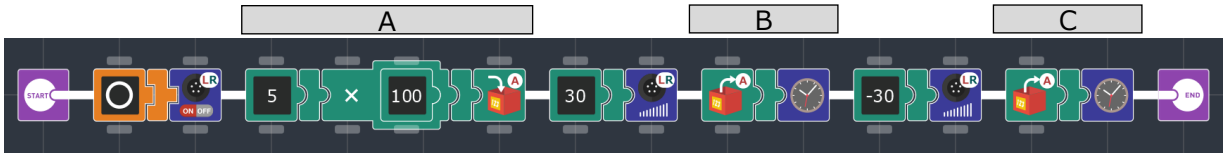
前進(0.5秒)→停止(0.5秒) →後進(0.5秒)→停止(0.5秒)を行う

このプログラムを作ってみると何となくわかるのですが、[5×100]の部分を4回も作るのはとても手間がかかります。また、「全ての秒数を0.8秒にしましょう」と言われたらたくさんパラメータを変えなければなりません。

第3章では、何度も使う数値を格納(しまい入れること)しておき、簡単に使い回すことができる「変数」という機能を学んでいきます。

前進(0.5秒)→後進(0.5秒)をしてみよう

下図のような「前進(0.5秒)→後進(0.5秒) 」するプログラムを作ります。



(例3) 前進(0.5秒)→後進(0.5秒)を行う

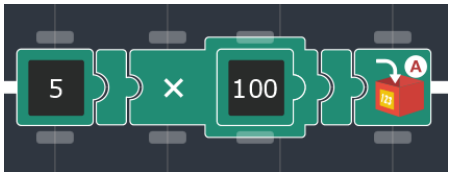
ここでは[数値変数 入力]と[数値変数 出力]のアイコンが使われています。「変数」とは、数値を一時的に格納することができる箱のようなものです。変数は、数値を入力して格納させることと、数値を出力して[数値定数]アイコンと同じように使うことができます。

A は数値を入力して記憶させている部分です。ここで[変数A]アイコンに5 × 100の計算結果[500]を入力することで、[500]を[変数A]に格納します。

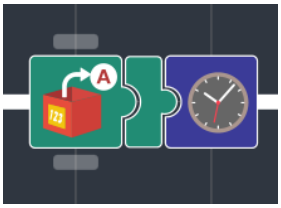
B は[変数A]の値を[待機]アイコンに渡している部分です。先程、A で[500]を[変数A]に格納したので、B では[待機]アイコンに[500]が渡されることになります。

そのため、B と C では「500ミリ秒(0.5秒)待機する」という処理が実行されます。

その結果、このプログラムをKOROBO2に転送すると、「前進(0.5秒)→後進(0.5秒) 」するプログラムが実行されます。



[A] 5 × 100 = 500を変数Aに記憶させる



[B][C] 変数A(500)を呼び出す

やってみよう

「前進(0.5秒)→後進(0.5秒) 」の動きでスピードの値[30]を[変数B]に格納し、[モータースピード]に[変数B]を使って値を渡して前進・後進させてみましょう。後進させるところでは[正負反転]アイコンを使うことで[-30]を渡すことができます。

前進(0.5秒)→後進(0.5秒)を行う

KOROBO3 応用編　第４章

同じ動きを繰り返そう

無限ループアイコンはありません

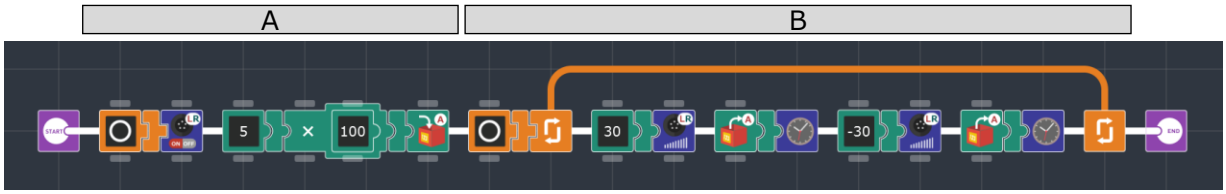
基本編(ベーシック)ではKOROBOのプログラムをずっと繰り返すときは[無限ループ]アイコンを使っていました。しかし、応用編(アドバンスト)には[無限ループ]アイコンはありません。そのかわりに[条件ループ]アイコンというものが使えるようになっています。

第4章では、[条件ループ]アイコンを使って、同じ動きをずっと繰り返したり、「壁にぶつかるまで繰り返す」といった動きをプログラミングしてみましょう。

[条件ループ]アイコン

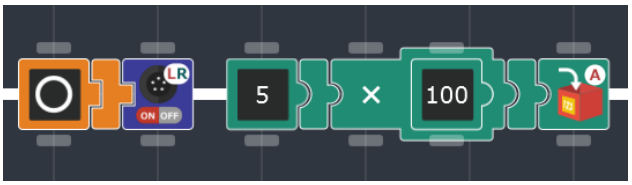
前進(0.5秒)→後進(0.5秒)をずっと繰り返そう

下図のような「前進(0.5秒)→後進(0.5秒) 」 をずっと繰り返すプログラムを作ります。



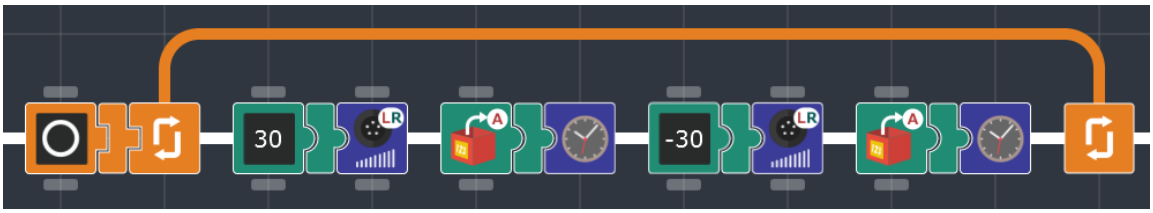
(例4) 前進(0.5秒)→後進(0.5秒)をずっと繰り返す

A はプログラムを動作させるときの準備を行っている部分です。繰り返す必要がない、最初に1回だけ行う処理(「初期化」とか「イニシャライズ」と呼ばれる)は A の部分に置きます。



[A] プログラムの初期化部分

このプログラムでは「モーターON」と「変数Aに0.5を格納」は最初に一度だけ処理すればよいので、初期化部分に入れています。
このプログラムのメインは B の部分です。

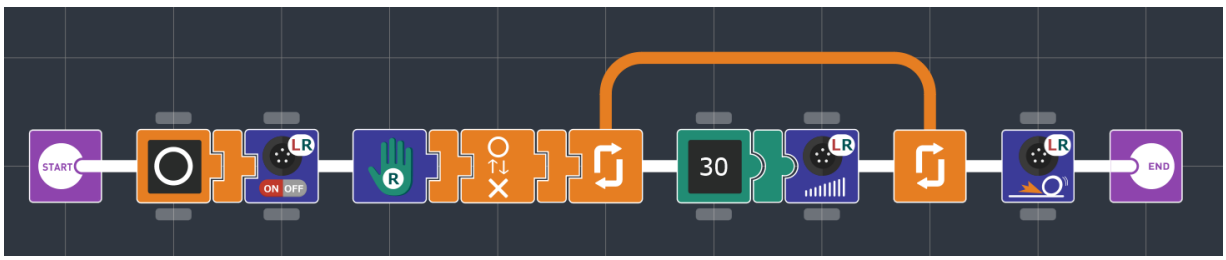


[B] プログラムのループ部分

[条件ループ]アイコンに真偽定数[○]が入力されています。[条件ループ]は「○を受け取っているとき、ループ内の処理を繰り返す」はたらきをするので、このプログラムではずっと真偽定数[○]が入力されているため、無限ループをすることになります。
KOROBOのプログラムを作るときに無限ループをさせることが多いと思いますが、そのときはこのプログラムのように[条件ループ]アイコンと[真偽定数]アイコンの[○]を使って無限ループを作りましょう。

やってみよう

「上ボディのタッチセンサー(TOUCH R)が押されるまでずっと前進する」という動きをするプログラムを作ってみましょう。 [条件ループ]アイコンは、[真偽定数]アイコンだけでなくセンサーの値も渡すことができます。
ここでループの条件は「TOUCH RがOFFのとき」となるので、[論理反転]アイコンを使います。



[B] プログラムのループ部分

KOROBO3 応用編　第５章

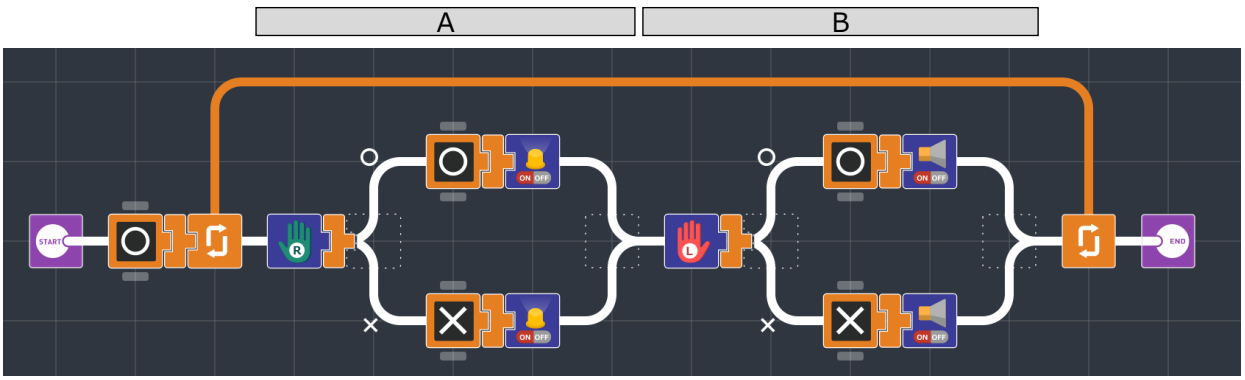
LED・ブザー・ボタンを使おう

プログラムの入念なチェックをするときに便利

これまで作ってきたKOROBOのプログラムは、モーターの動作を確認することでうまく動いているかどうかをチェックしてきました。
KOROBOに限らずロボットを動かすときは、モーターを動かさずにプログラムの誤りを修正する作業(プログラミングの世界では**「デバッグ」**といいます)をしたいことがよくあります。
第５章では、モーターを使わずにセンサーが反応しているかをチェックする、上手なデバッグ方法を学んでいきましょう。

LEDとブザーでタッチセンサーをチェックしよう

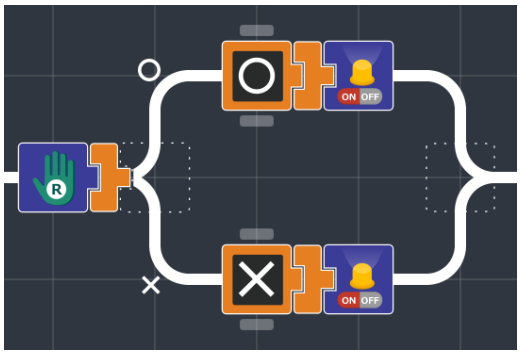
下図のような「タッチセンサー(TOUCH L, TOUCH R)をチェックする」プログラムを作ります。



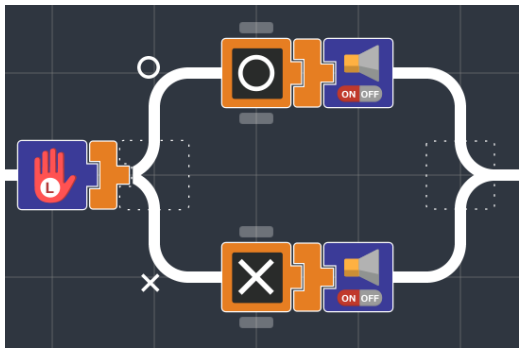
(例5) タッチセンサーをチェックする

A は、「TOUCH R(上ボディのタッチセンサー)が押されたらLEDを点灯、押されなかったらLEDを消灯」を行い、B は「TOUCH L(下ボディのタッチセンサー)が押されたらブザーを鳴らす、押されな

かったらブザーを鳴らさないという処理を行うプログラムです。
[汎用LED ON/OFF]アイコンに[○]を渡すとLEDが点灯、[×]を渡すとLEDが消灯します。[ブザー ON/OFF]アイコンに[○]を渡すとブザーが鳴り、[×]を渡すとブザーが止まります。



[A] 汎用LED ON/OFF



[B] ブザー ON/OFF

このようにLEDとブザーを使うことにより、モーターを動かさなくてもセンサーが反応しているかをチェックすることができます。

やってみよう

タッチセンサー2個をそれぞれLEDとブザーで条件分岐を使ってチェックをするプログラムを作成しましたが、それと同じ動きをするプログラムを「[条件分岐]アイコン無しで」作ってみましょう。
下図のようにプログラムを作ると、(例5)のプログラムと同じような動きをします。このように、LEDやブザーなどの真偽値を受け取るアイコンはセンサーの真偽値を直接渡すこともできるのです。

[条件分岐]アイコン無しでタッチセンサーのチェックをする

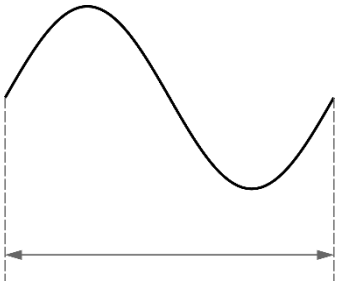
KOROBO3 応用編　第６章

ブザーの周波数を変えてみよう

音の周波数とは

KOROBOのブザーは音の高さを変えて出すことができます。この音の高さを数値で表したものを音の「周波数」といいます。「ド・レ・ミ…」と音階が高くなるにつれて周波数は高くなります。音は図のように波の性質を持っており、1秒間あたりに波が振動する回数が周波数の値(単位はHz:ヘルツ)となります。

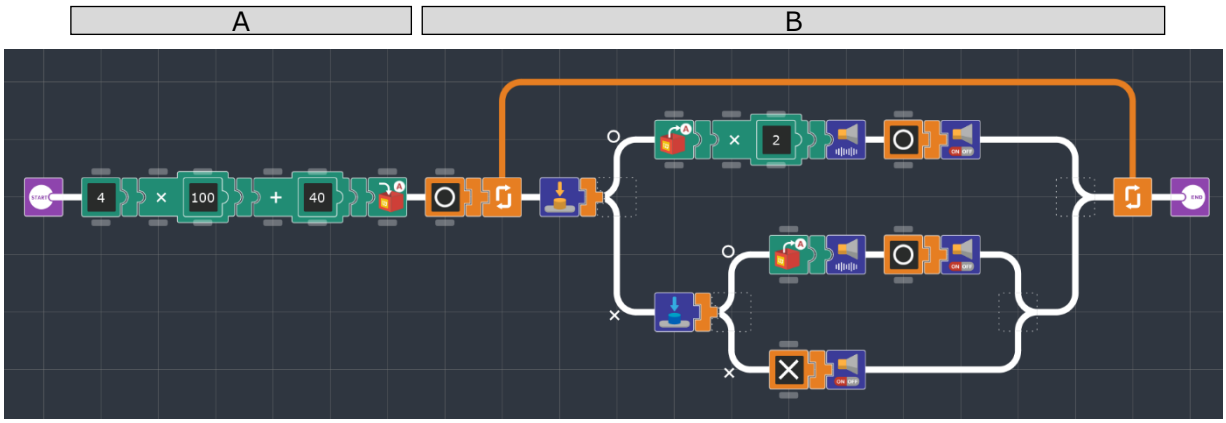
特に音の周波数の「440Hz」は基準周波数と呼ばれ、ラジオの時報などでも使われています。またピアノ鍵盤の中央に近いオクターブの「ラ」の音はこの基準周波数440Hzに合わせています。



1秒間あたりに波が振動する回数が周波数(単位:Hz)

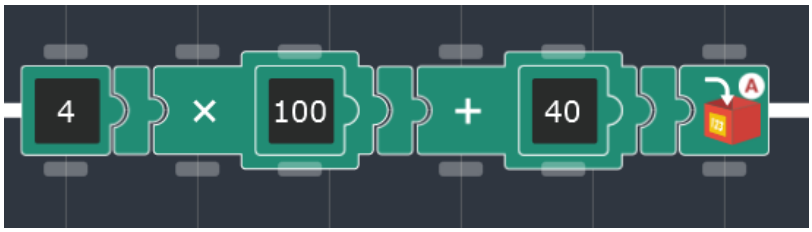
スイッチで音を鳴らしわけよう

下図のような「黄スイッチ(汎用ボタン)を押したら2倍の880Hz、青スイッチ(スタートスイッチ)を押したら440Hzをブザーで鳴らし、どちらも押さなかったらブザーを鳴らさない」プログラムを作ります。



(例6) ブザーの周波数を変えて鳴らす

A は数値[440]を[変数A]アイコンに渡して格納しています。これが基準周波数となります。

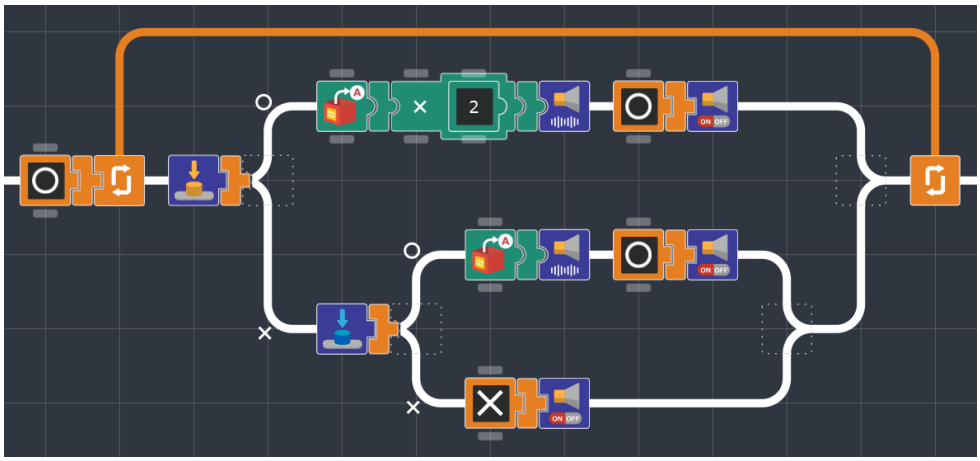


[A] 基準周波数440Hzの格納

B は黄色と青色のスイッチの状態を確認して、

- 黄スイッチ(BUTTON)が押されていたら$440 \times 2 = 880$Hzを鳴らす
- 青スイッチ(START)が押されていたら440Hzを鳴らす
- どちらのスイッチも押されていなかったらブザーを鳴らさない

という動作をします。この880Hzは、基準周波数の「ラ」より1オクターブ高い「ラ」の音です。



[B] 440Hzと880Hzをボタン操作で鳴らす

黄スイッチ(BUTTON)と青スイッチ(START)はKOROBOの基板上にあるプッシュスイッチです。この2つのスイッチはプログラム上でON/OFFを検知することができ、センサーの代わりに条件分岐による動作チェックにも使えます。

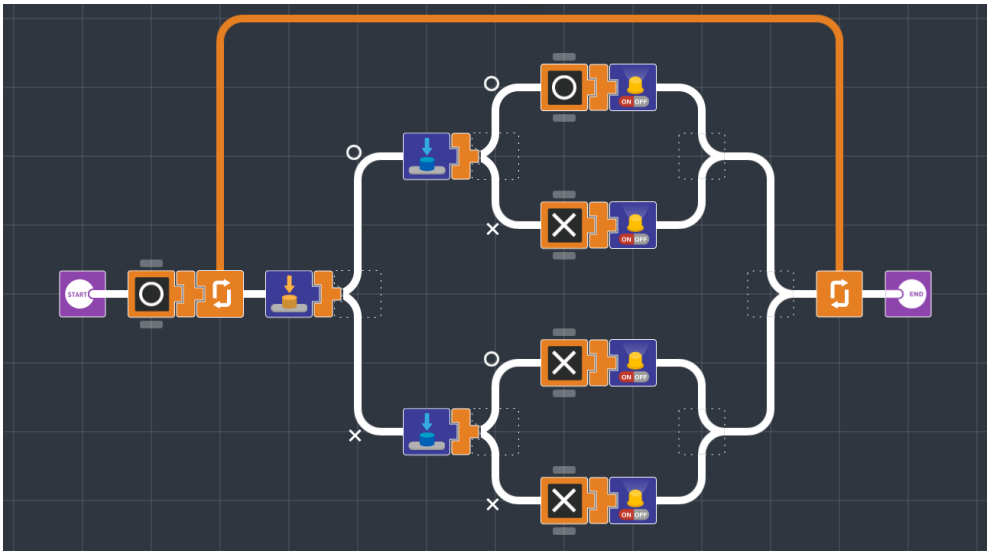
黄スイッチ
(BUTTON)

青スイッチ
(START)

論理演算をしてみよう

入れ子の条件分岐をコンパクトにまとめたい

「黄と青のスイッチ両方が押された場合、LEDを点灯。それ以外の場合、LEDを消灯」というプログラムを下図のように作ってみました。



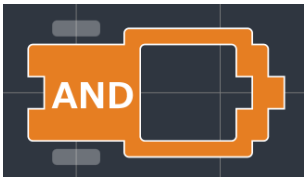
黄と青のスイッチ両方が押されたらLEDを点灯、それ以外は消灯

もちろんこれで正しく動きますが、第7章では**[論理演算]アイコン**を使ってこれと同じ動きのプログラムをコンパクトに作っていきます。

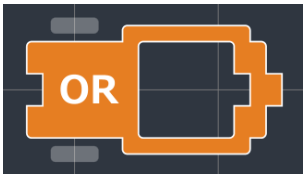
[論理演算]アイコンはパラメータボタンを押してはたらきを変えることができます。呼び方はそれぞれAND(アンド), OR(オア), XOR(エックスオア)と呼びます。

[論理演算]アイコンは、アイコンの左側と内側それぞれに真偽値を入力として渡して、その演算結果が右側に出力されるはたらきをします。

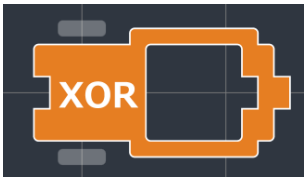
AND, OR, XORはそれぞれで演算の内容が変わってきます。



AND(アンド)



OR(オア)



XOR(エックスオア)

論理演算アイコンを使ってみよう

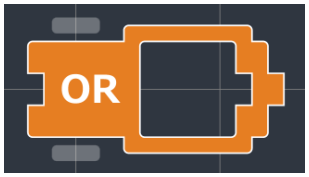
AND, OR, XORがそれぞれどのようなはたらきをするか見てみましょう。
入力される2つの真偽値をそれぞれ[入力A][入力B]とすると、[AND], [OR], [XOR]は下表のような入力と出力の関係となります。

AND

[A]と[B]の両方が ○ の場合、出力が ○ になる。
それ以外は ✕ になる。

入力A	入力B	出力(AND)
○	○	○
○	✕	✕
✕	○	✕
✕	✕	✕

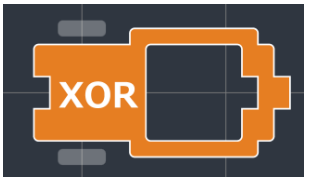
OR



どちらか一方でも ○ の場合、出力が ○ になる。
それ以外は ✕ になる。

入力A	入力B	出力(OR)
○	○	○
○	✕	○
✕	○	○
✕	✕	✕

XOR



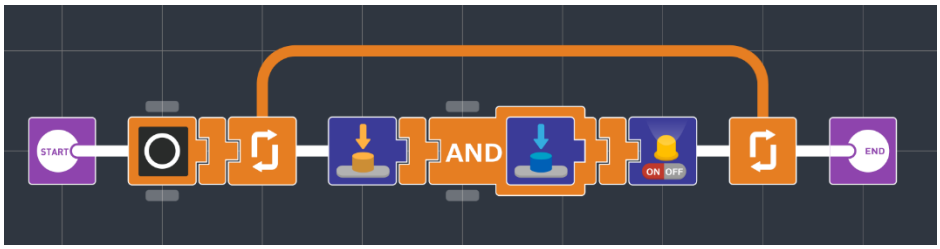
片方が ○ でもう片方が ✕ の場合、出力が ○ になる。両方同じときは ✕ になる。

入力A	入力B	出力(XOR)
○	○	✕
○	✕	○
✕	○	○
✕	✕	✕

この章の最初に示した**「黄と青のスイッチ両方が押された場合、LEDを点灯。それ以外の場合、LEDを消灯」**は先程の [AND], [OR], [XOR]で見ると、[AND]の

[A]と[B]の両方が ○ の場合、出力が ○ になる。それ以外は ✕ になる。
＝ ＝ ＝
黄と青のスイッチ両方が押された　LEDが点灯　LEDが消灯

が同じように対応していることがわかります。
実際に[AND]アイコンを使ってプログラミングしてみると、(例7)のプログラムなります。条件分岐を使うよりコンパクトになりました。



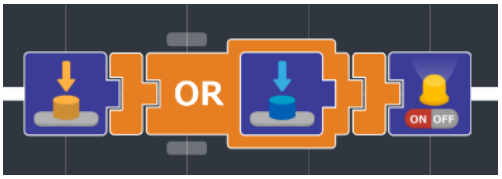
(例7) 黄と青のスイッチ両方が押されたらLEDを点灯、それ以外は消灯

また、[論理反転]アイコンは論理演算「NOT(ノット)」のはたらきを持つアイコンで、これは入力の[○]と[✕]を反転させた真偽値を出力します。

[論理反転]アイコン

やってみよう

先程の(例7)のプログラムでは[AND]アイコンを使いましたが、これを[OR]または [XOR]に変えてみてそれぞれプログラムを実行し、スイッチの状態とLEDの点灯・消灯の関係が前ページの表と同じになるかをチェックしてみましょう。



ORにした場合

XORにした場合

KOROBO3 応用編　第８章

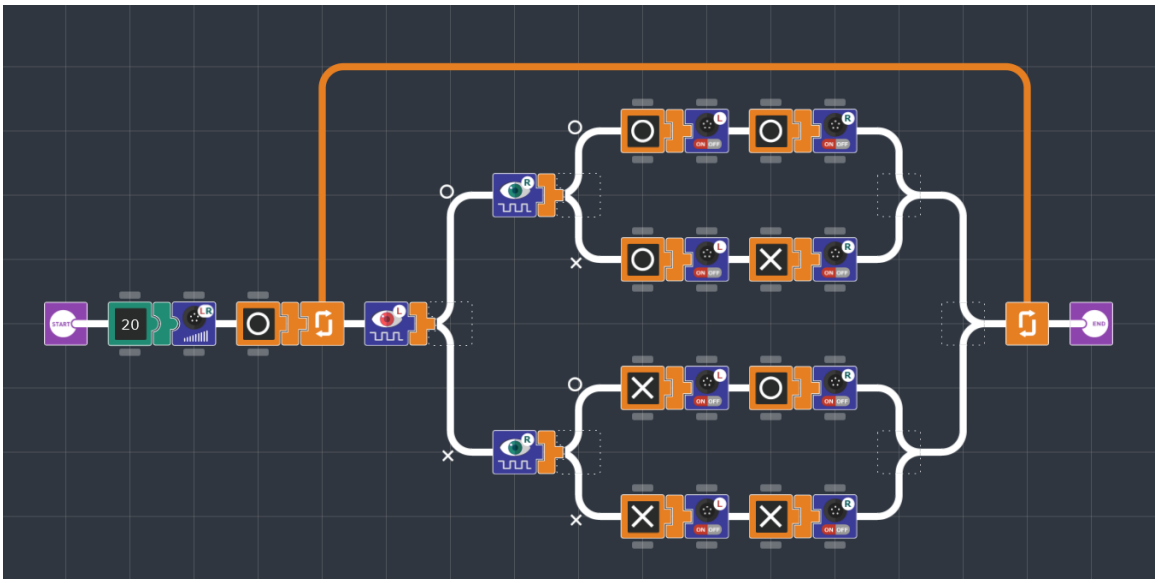
ライントレース(応用編)

アドバンストでライントレースをしてみよう

第8章では、基本編でも作成したライントレースのプログラムを作成します。ベーシックとアドバンストとでは使えるアイコンが変わっていますが、アドバンストではモーターのスピードを細かく設定することができるため、動き方を自由に変えることができるようになります。決められたコースをいかに早く、かつ確実にゴールすることができるか、前進のスピードや旋回の方法を工夫しながら、より高性能なライントレースができるKOROBOを作りましょう。

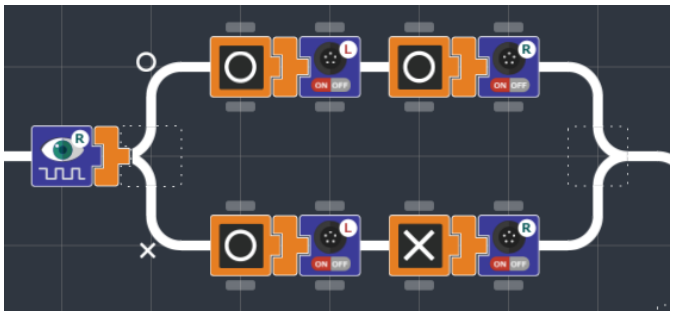
数値や真偽値でコントロールしよう

アドバンストで作成したライントレースのプログラムを下図のように作ります。ここでは最初にスピード設定をして、ループ内でモーターを[動かす]または[止める]の選択をしています。ラインから飛び出す場合は最初のスピード設定の値を小さくしてみましょう。

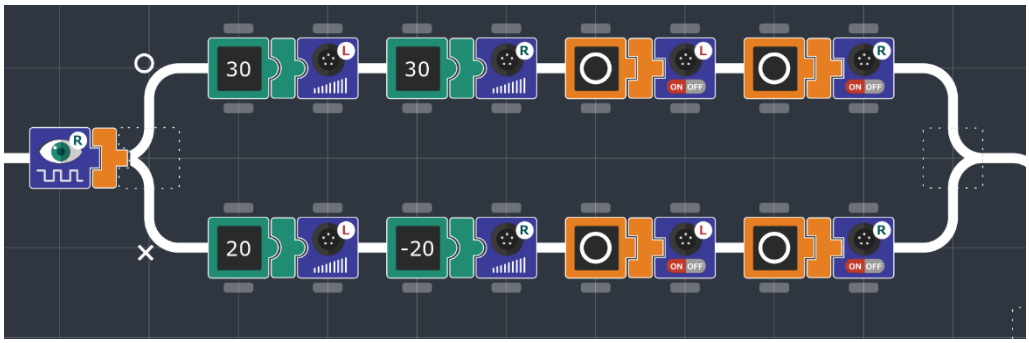


(例8) 光センサー2個を使ってライントレースをする

(例8)のプログラムでは下図[A]のように左右のモーターを[動かす]もしくは[止める]のみをプログラムしていますが、下図②のようにそれぞれの動きで細かくスピードを設定することでより良いライントレースの動きを実現できるかもしれません。



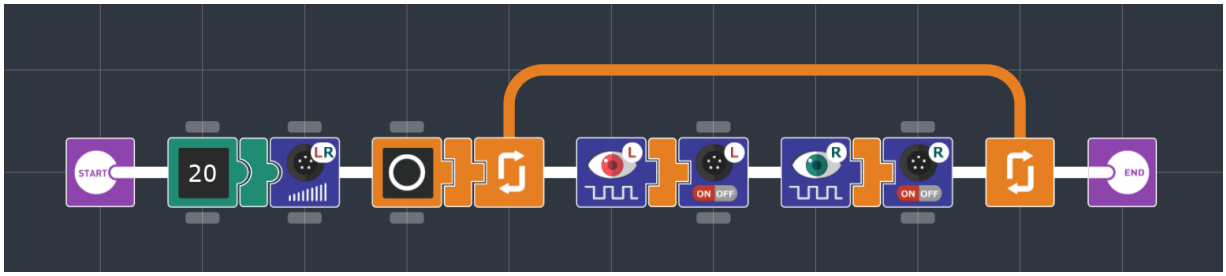
① 動かす もしくは 止める のみ



② 細かくスピードを設定

やってみよう

(例8)のプログラムの入れ子になっている条件分岐は、下図のプログラムのように光センサー(左)をモーター(左)に、光センサー(右)をモーター(右)に値を渡すように対応させてライントレースをさせることもできます。試しにこのプログラムでライントレースをさせてみて、なぜこのように動くのか考えてみましょう。



条件分岐を使わずにライントレースをする

KOROBO3応用編

アイコン一覧 (KOROBO3 アドバンスト)

アイコン	アイコンの名前・説明
	**＜条件分岐＞** 真偽値を受け取り、[○]を受け取った場合は上の経路へ、[×]を受け取った場合は下の経路へ処理を進めます。 ※「条件分岐」アイコンはデータ(真偽値)を受け取るアイコンであるため、その左側には必ずデータ(真偽値)を渡すアイコンが配置されなくてはいけません。
	**＜汎用LED ON/OFF＞** 真偽値を受け取り、基板上のLED4を点灯・消灯させます。[○]が与えられると点灯し、[×]が与えられると消灯します。
	**＜光センサーLED ON/OFF＞** 真偽値を受け取り、光センサー投光用LEDを点灯・消灯させます。[○]が与えられると点灯し、[×]が与えられると消灯します。 パラメータボタンによって光センサーのON/OFF切り替えを行う対象を左(L)・右(R)・左右両方(LR)から選択することができます。 L：光センサー(左)の投光用LEDを点灯/消灯 R：光センサー(右)の投光用LEDを点灯/消灯 LR：左右両方の光センサーの投光用LEDを同時に点灯/消灯

アイコン	アイコンの名前・説明
	**<ブザー ON/OFF>** 真偽値を受け取り、ブザーのON/OFF状態を切り替えます。 [○]を受け取るとON、[×]を受け取るとOFFとなります。
	**<ブザー 周波数>** 数値を受け取り、ブザーの音の周波数を設定します。
	**<モーター ON/OFF>** 真偽値を受け取り、モーターのON/OFF状態を切り替えます。 パラメータボタンによってモーターのON/OFF切り替えを行う対象を左(L)・右(R)・左右両方(LR)から選択することができます。 [○]を受け取ると回転、[×]を受け取ると停止します。 L：左モーターを回転/停止 R：右モーターを回転/停止 LR：左右両方のモーターを同時に回転/停止
	**<モータースピード>** 数値を受け取り、モーターの回転速度を設定します。 プラスの数値を受け取った場合は正転速度として、マイナスの数値を受け取った場合は逆転速度として扱います。 受け取る値の範囲は最小-100、最大100となっており、範囲外の値を受け取った場合は、-100あるいは100に切り捨てます。 受け取る値が0のときモーターの回転は停止します。 [モーター ON/OFF]アイコンと同様に対象をL / R / LRから選択できます。
	**<モーターブレーキ>** モーターにブレーキをかけて停止させます。 パラメータボタンによってモーターのブレーキを行う対象を左(L)・右(R)・左右両方(LR)から選択することができます。 L：左モーターをブレーキ R：右モーターをブレーキ LR：左右両方のモーターを同時にブレーキ
	**<待機 ミリ秒>** 数値を受け取り、数値のミリ秒間(1000分の1秒間) 、待機します。 例：[100]を受け取ると、$\frac{1}{1000} \times 100 = 0.001 \times 100 = 0.1$ となり、0.1秒間待機します。

アイコン	アイコンの名前・説明
 	**<タッチセンサー (左 および 右)>** タッチセンサーの状態を読み取り、真偽値で次のアイコンに渡します。タッチセンサーのスイッチが押されている場合は[○]が、押されていない場合は[×]が次のアイコンに渡されます。 アイコンのイラストに「L」と書かれているものが「TOUCH L」、「R」と書かれているものが「TOUCH R」のタッチセンサーです。
 	**<光センサー (左 および 右)>** 光センサーの状態を読み取り、真偽値で次のアイコンに渡します。光センサーがON状態(光センサーに光が十分に当たっている)の場合は[○]が、OFF状態(光センサーに光が十分に当たっていない)の場合は[×]が次のアイコンに渡されます。 アイコンのイラストに「L」と書かれているものが「PHOTO L」、「R」と書かれているものが「PHOTO R」の光センサーです。
	**<ボタン 青>** 基板上の青ボタン(START)が押されているか押されていないかの状態を真偽値で右のアイコンに渡します。ボタンが押されている場合は[○]が、ボタンが押されていない場合は[×]が渡されます。
	**<ボタン 黄>** 基板上の黄ボタン(BUTTON)が押されているか押されていないかの状態を真偽値で右のアイコンに渡します。ボタンが押されている場合は[○]が、ボタンが押されていない場合は[×]が渡されます。
	**<ブレーク>** 後に続くループ内の処理を中断して、ループ末尾の次の処理へ移ります。ブレークはループを中断させる目的のアイコンであるため、必ずループの内側に配置する必要があります。
	**<コンティニュー>** 後に続くループ内の処理を中断して、ループ先頭の処理へ移ります。コンティニューはループを中断させるアイコンであるため、必ずループの内側に配置する必要があります。

アイコン	アイコンの名前・説明
	**＜条件ループ＞** 真偽値を受け取り、[○]を受け取った場合は、[条件ループ]内の処理を実行します。ループの最後に到達した場合は、[条件ループ]の先頭に戻り、もう一度真偽値を受け取ります。[×]を受け取った場合は、[条件ループ]末尾の次のアイコンに移動します。
	**＜真偽定数＞** パラメータボタンで設定した真偽値[○]または[×]を右のアイコンに渡します。
	**＜真偽変数 入力＞** 真偽値を受け取り、パラメータボタンで指定した真偽変数(A, B, C, D, E)に格納します。
	**＜真偽変数 出力＞** パラメータボタンで指定した真偽変数(A, B, C, D, E)から真偽値を取り出して、右のアイコンに渡します。なお各真偽変数の初期値は[×]です。
	**＜論理反転＞** 真偽値を受け取り、その値を反転させた真偽値を右のアイコンに渡します。[○]を受け取ると[×]を渡し、[×]を受け取ると[○]を渡します。
	**＜論理演算＞** [論理演算]アイコンの左側と内側で真偽値をそれぞれ受け取り、左側と内側の真偽値で論理演算を行った結果を、真偽値で右のアイコンに渡します。論理演算の種類(AND, OR, XOR)は、パラメータボタンで選択します。

アイコン	アイコンの名前・説明
	＜回数ループ＞ 数値を受け取り、その回数、[回数ループ]内の処理を実行します。
	＜数値定数＞ パラメータボタンで設定した数値を右のアイコンに渡します。パラメータボタンをクリックするごとに数値は1ずつ増減します(パラメータボタン(上)は増加、パラメータボタン(下)は減少)。またパラメータボタンに対してドラッグ操作を行うと、目盛りが表示されて、数値の設定をドラッグで行うことができます。設定できる数値は最小-100、最大100の範囲内の整数です。
	＜数値変数 入力＞ 数値を受け取り、パラメータボタンで指定した数値変数(A, B, C, D, E)に格納します。格納できる数値の範囲は、-32768 ～ 32767の整数となります。
	＜数値変数 出力＞ パラメータボタンで指定した数値変数(A, B, C, D, E)から数値を取り出して、右のアイコンに渡します。なお各数値変数の初期値は0です。
	＜算術演算＞ [算術演算]アイコンの左側と内側で数値をそれぞれ受け取り、左側と内側の数値で算術演算を行った結果を、数値で右のアイコンに渡します。算術演算の種類(足し算、引き算、掛け算、割り算)は、パラメータボタンで選択します。なお割り算で内部のアイコンの数値が「0」のときは、実行時に「1」に置き換えられます。
	＜正負反転＞ 数値を受け取り、その値の正負(プラスマイナス)を反転させた数値を右のアイコンに渡します。
	＜算術比較＞ [算術比較]アイコンの左側と内側で数値をそれぞれ受け取り、左側と内側の数値で算術比較を行った結果を、真偽値で右のアイコンに渡します。算術比較の種類(大なり(>), 小なり(<), 等価(=))は、パラメータボタンで選択します。