

2010/10/22

Msako 技術資料

メッセージプログラミングガイド

ユーザは、次に示すメッセージを使用して Msako と通信する独自のアプリケーションプログラムを開発することができる。

- 接続メッセージ
- 動体検知メッセージ
- 開始・停止指令メッセージ
- データメッセージ配信予約メッセージ
- 基本情報取得メッセージ
- パラメータ取得メッセージ
- パラメータ設定メッセージ
- データメッセージ
- バージョン取得メッセージ
- 再構成メッセージ
- 再起動メッセージ
- 外部機器警報メッセージ

「接続メッセージ」、「動体検知メッセージ」、「開始・停止指令メッセージ」、「データメッセージ配信予約メッセージ」、「バージョン取得メッセージ」、「再構成メッセージ」、「再起動メッセージ」、「外部機器警報メッセージ」は、**WM_MSAKO_**で始まる Msako 独自のメッセージを使用する。ユーザのアプリケーションプログラムは、これらのメッセージを使用する場合、あらかじめ **RegisterWindowMessage** を使って使用するメッセージを登録して、メッセージ ID を取得しておく必要がある。**RegisterWindowMessage** に渡す引数としては、次のように、シンボルと等しい値の文字列を指定する。

WM_MSAKO_XXX メッセージ登録の例 (C++の場合)

```
UINT WM_MSAKO_MOTION_DETECTED = RegisterWindowMessage(_T("WM_MSAKO_MOTION_DETECTED"));
```

WM_MSAKO_で始まる Msako 独自のメッセージを受信する場合は、次の例のようにメッセージハンドラの中で処理する。

WM_MSAKO_XXX メッセージ受信の例 (C++の場合)

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_MSAKO_MOTION_DETECTED:
            // ユーザ独自の処理をここに書く
            break;
    }
}
```

WM_MSAKO_で始まる Msako 独自のメッセージを Msako へ送信する場合は、**PostMessage** または **SendMessage** を使用する。これらの関数に渡す引数のうち、送信先のウィンドウのハンドルとして、Msako のウィンドウハンドルが必要となる。ユーザのアプリケーションプログラムは「接続メッセージ」を使って Msako のウィンドウハンドルを知ることができる。**HWND_BROADCAST** を使用することも可能だが、**HWND_BROADCAST** を指定した場合は、メッセージはシステム内のすべてのトップレベルウィンドウ (親を持たないウィンドウ) へ届けられるので注意が必要だ。Msako のウィンドウハンドルは **FindWindow** などで調べることもできる。また、Msako が起動すると **WM_MSAKO_INITIALIZED** メッセージが配信されるので、それによっても Msako のウィンドウハンドルを知ることができる。**PostMessage** または **SendMessage** の **WPARAM** 引数には、送信元であるユーザのアプリケーションプログラムのウィンドウハンドルを指定する。**LPARAM** 引数に指定する値は、メッセージによって異なる。

WM_MSAKO_XXX メッセージ送信の例 (C++の場合)

```
BOOL result = PostMessage(hMsako, WM_MSAKO_START, hWnd, NULL);
```

一方、「基本情報取得メッセージ」、「パラメータ取得メッセージ」、「パラメータ設定メッセージ」、「データメッセージ」は、システムで既定の **WM_COPYDATA** メッセージを使用する。Msako が送信した **WM_COPYDATA** メッセージを受信する場合は、次の例のようにメッセージハンドラの中で処理する。この場合、自分が送信したメッセージが自分自身に届く場合もあるので、メッセージの処理がループしないよう、受信したメッセージの送信元のウィンドウが自分自身ではないことを確認する必要がある。**WM_COPYDATA** メッセージを受信した時、**WPARAM** には送信元のウィンドウのハンドルが、**LPARAM** には受信した **COPYDATASTRUCT** 構造体へのポインタが設定されている。ユーザのアプリケーションプログラムは、受取った **COPYDATASTRUCT** 構造体の **dwData** メンバを調べ、それが Msako から送信されたメッセージであることを確認した上で処理すること。

WM_COPYDATA メッセージ受信の例 (C++の場合)

```

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    HWND hPeer = (HWND)wParam;
    COPYDATASTRUCT *data = (COPYDATASTRUCT *)lParam;
    switch(msg)
    {
        case WM_COPYDATA:
            if(hWnd != hPeer)
            {
                switch(data->dwData)
                {
                    case MM_GET_PARAM:
                        // ユーザ独自の処理をここに書く
                        break;
                }
            }
            break;
    }
}

```

ユーザのアプリケーションプログラムは、WM_COPYDATA メッセージを Msako へ送信する場合、必ず **SendMessage** を使用すること。**SendMessage** の送信先のウィンドウのハンドルとして、Msako のウィンドウハンドルが必要となる。ユーザのアプリケーションプログラムは「接続メッセージ」を使って Msako のウィンドウハンドルを知ることができる。HWND_BROADCAST を使用することも可能だが、HWND_BROADCAST を指定した場合は、メッセージはシステム内のすべてのトップレベルウィンドウ(親を持たないウィンドウ)へ届けられるので注意が必要だ。Msako のウィンドウハンドルは **FindWindow** などで調べることもできる。また、Msako が起動すると WM_MSAKO_INITIALIZED メッセージが配信されるので、それによっても Msako のウィンドウハンドルを知ることができる。**SendMessage** の WPARAM 引数には、送信元であるユーザのアプリケーションプログラムのウィンドウハンドルを指定し、LPARAM 引数には、送信する COPYDATASTRUCT 構造体へのポインタを指定する。送信する COPYDATASTRUCT 構造体の内容は、送信するメッセージによって異なる。

WM_COPYDATA メッセージ送信の例 (C++の場合)

```

COPYDATASTRUCT data;
data.dwData = MM_GET_DEVICE_NAME;
data.cbData = 0;
data.lpData = NULL;
BOOL result = SendMessage(hMsako, WM_COPYDATA, hWnd, &data);

```

Msako のメッセージに関する種々の定義は次のヘッダファイルに記述されている。C++以外の言語でアプリケーションプログラムを作成する場合、これらのヘッダファイルを参考にして同等の定義をするといいたいだろう。

```

MsakoMsg.h
MsakoTarget.h
MsakoStatus.h

```

ユーザが、メッセージを使ってMsakoと通信するアプリケーションプログラムを作成するうえで、必要となるシステムの関数や関連する構造体などについての情報は、次の URL から入手することができる。

RegisterWindowMessage 関数

<http://msdn.microsoft.com/ja-jp/library/cc410981.aspx>

SendMessage 関数

<http://msdn.microsoft.com/ja-jp/library/cc411022.aspx>

PostMessage 関数

<http://msdn.microsoft.com/ja-jp/library/cc410952.aspx>

FindWindow 関数

<http://msdn.microsoft.com/ja-jp/library/cc364634.aspx>

WM_COPYDATA メッセージ

[http://msdn.microsoft.com/en-us/library/ms649011\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms649011(VS.85).aspx)

COPYDATASTRUCT 構造体

[http://msdn.microsoft.com/en-us/library/ms649010\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms649010(v=VS.85).aspx)

接続メッセージ

このメッセージは、Msako とユーザのアプリケーションプログラムがメッセージを交換するにあたり、お互いのウィンドウハンドルを知り合うために使う。このメッセージは、要求とそれに対する応答という 1 往復のメッセージ交換で完結する。まず、ユーザのアプリケーションプログラムは LPARAM に発見したいモジュール種別として MD_MSAKO を設定し、要求のメッセージを送信する。このとき、送信先のウィンドウハンドルは、はまだ不明であるので、HWND_BROADCAST (すべてのトップレベルウィンドウ) を指定する。このメッセージを受取った Msako は、要求元のユーザのアプリケーションプログラムへ応答のメッセージを送信する。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_DISCOVER	モジュール発見 (要求)	送信元ウィンドウハンドル	モジュール種別	User→Msako
WM_MSAKO_DISCOVERED	モジュール発見 (応答)	送信元ウィンドウハンドル	モジュール種別	Msako→User

モジュール種別

値	意味
MD_ALL	すべてのモジュール
MD_MSAKO	Msako 本体
MD_CLIENT	ユーザのアプリケーションプログラム

動体検知メッセージ

このメッセージは、動体検知に伴って Msako からシステム内のすべてのトップレベルウィンドウ (親を持たないウィンドウ) へ送信される。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_MOTION_DETECTED	フレームで動体を検知した	送信元ウィンドウハンドル	フレーム番号	Msako→User
WM_MSAKO_EVENT_START	イベントが開始した	送信元ウィンドウハンドル	イベント番号	Msako→User
WM_MSAKO_EVENT_END	イベントが終了した	送信元ウィンドウハンドル	イベント番号	Msako→User
WM_MSAKO_STARTED	動体検知を開始した	送信元ウィンドウハンドル	NULL	Msako→User
WM_MSAKO_STOPPED	動体検知を停止した	送信元ウィンドウハンドル	NULL	Msako→User
WM_MSAKO_INITIALIZED	Msako が起動した	送信元ウィンドウハンドル	NULL	Msako→User
WM_MSAKO_TERMINATED	Msako が停止した	送信元ウィンドウハンドル	NULL	Msako→User

開始・停止指令メッセージ

このメッセージは、ユーザのアプリケーションプログラムが Msako の動体検知の開始・停止を制御するためのメッセージである。このメッセージを受信すると、Msako は指令に従って動体検知を開始または停止する。このメッセージを送信するにあたり、ユーザのアプリケーションプログラムは送信先の Msako のウィンドウハンドルが必要となる。このために、「接続メッセージ」を使えば、あらかじめ Msako のウィンドウハンドルを入手することができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_START	動体検知を開始せよ	送信元ウィンドウハンドル	NULL	User→Msako
WM_MSAKO_STOP	動体検知を停止せよ	送信元ウィンドウハンドル	NULL	User→Msako

データメッセージ配信予約メッセージ

このメッセージは、ユーザのアプリケーションプログラムが Msako からデータメッセージの配信が受けられるように、あらかじめ事前に予約しておくためのメッセージである。このメッセージを受信した Msako は、このメッセージを送信したユーザのアプリケーションプログラムをデータメッセージ配信の対象として登録する。このメッセージを送信するにあたり、ユーザのアプリケーションプログラムは送信先の Msako のウィンドウハンドルが必要となる。このために、「接続メッセージ」を使えば、あらかじめ Msako のウィンドウハンドルを入手することができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_LISTEN	データメッセージを配信せよ	送信元ウィンドウハンドル	NULL	User→Msako
WM_MSAKO_UNLISTEN	データメッセージの配信を停止せよ	送信元ウィンドウハンドル	NULL	User→Msako

基本情報取得メッセージ

このメッセージは、ユーザのアプリケーションプログラムが Msako の基本情報を取得するためのメッセージである。まず、ユーザアプリケーションプログラムが要求のメッセージを Msako へ送信する。要求のメッセージを受信した Msako は、応答のメッセージを要求元のユーザのアプリケーションプログラムへ送信する。このメッセージを送信するにあたり、ユーザのアプリケーションプログラムは送信先の Msako のウィンドウハンドルが必要となる。このために、「接続メッセージ」を使えば、あらかじめ Msako のウィンドウハンドルを入手することができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_COPYDATA	基本情報を取得する(要求)	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	User→Msako
WM_COPYDATA	基本情報を取得する(応答)	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	Msako→User

このメッセージで使用する COPYDATASTRUCT 構造体は次のように設定する。ただし、ユーザのアプリケーションプログラムが要求を送信するときは、cbData と lpData は設定する必要はない。

COPYDATASTRUCT 構造体の設定

dwData	cbData	lpData	意味
MM_GET_DEVICE_NAME	カメラ名の長さ	カメラ名へのポインタ	カメラの名称を取得する
MM_GET_ALIAS	カメラ別名の長さ	カメラ別名へのポインタ	カメラの別名を取得する
MM_GET_LOCATION	撮影場所名の長さ	撮影場所名へのポインタ	撮影場所を取得する
MM_GET_CAMERA_ID	カメラ識別番号の長さ	カメラ識別番号へのポインタ	カメラの識別番号を取得する
MM_GET_IMAGE_SIZE	MM_SIZE_DATA 構造体の大きさ	MM_SIZE_DATA 構造体へのポインタ	画像の大きさを取得する
MM_GET_STATUS	MM_STATUS_DATA 構造体の大きさ	MM_STATUS_DATA 構造体へのポインタ	動体検知の状態を取得する

MM_SIZE_DATA 構造体の定義

```
typedef struct
{
    int width;
    int height;
    int rotate;
}
MM_SIZE_DATA, *PMM_SIZE_DATA;
```

width

画像の横幅(ピクセル)

height

画像の高さ(ピクセル)

rotate

画像回転角 (Degree)

MM_STATUS_DATA 構造体の定義

```
typedef struct
{
    MSAKO_STATUS status;
    int cputime;
}
MM_STATUS_DATA, *PMM_STATUS_DATA;
```

status

Msako の状態

status の値	意味
ST_STOP	停止中
ST_INIT_DEVICE	カメラ初期化中
ST_TUNING	チューニング中
ST_REDY	監視中
ST_MOTION_DETECTED	動体を検知

cputime

処理時間 (ミリ秒)

パラメータ取得メッセージ

このメッセージは、ユーザのアプリケーションプログラムが Msako のパラメータの値を取得するためのメッセージである。まず、ユーザのアプリケーションプログラムが要求のメッセージを Msako へ送信する。要求のメッセージを受信した Msako は、応答のメッセージとして、要求されたパラメータの値を要求元のユーザのアプリケーションプログラムへ送信する。このメッセージを送信するにあたり、ユーザのアプリケーションプログラムは送信先の Msako のウィンドウハンドルが必要となる。このために、「接続メッセージ」を使えば、あらかじめ Msako のウィンドウハンドルを入手することができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_COPYDATA	パラメータを取得する(要求)	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	User→Msako
WM_COPYDATA	パラメータを取得する(応答)	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	Msako→User

このメッセージで使用する COPYDATASTRUCT 構造体は次のように設定する。ただし、ユーザのアプリケーションプログラムが要求を送信するときは、MM_PARAM_DATA 構造体の param メンバを設定する必要はない。

COPYDATASTRUCT 構造体の設定

dwData	cbData	lpData	意味
MM_GET_PARAM	MM_PARAM_DATA 構造体の大きさ	MM_PARAM_DATA 構造体へのポインタ	パラメータの値を取得する

MM_PARAM_DATA 構造体の定義

```
typedef struct
{
    MM_PARAM_TYPE type;
    union
    {
        BYTE byte_value;
        WORD word_value;
        UINT uint_value;
        int int_value;
        long long_value;
        float float_value;
        double double_value;
    }
    param;
}
MM_PARAM_DATA, *PMM_PARAM_DATA;
```

type

パラメータの種類

type の値	意味	データ型	範囲
MP_SENSE_LEVEL	検出レベル(%)	符号付整数	0 以上 100 以下
MP_FRAME_RATE	サンプル速度 (fps x 10)	符号付整数	5 以上 300 以下
MP_BLOCK_SIZE	ブロックの大きさ *	符号付整数	8、16、24、32
MP_NUM_BUFFER	バッファ数 *	符号付整数	1 以上 64 以下
MP_NUM_TARGET	ターゲット数	符号付整数	1 以上 3 以下
MP_FLASH_COMP	閃光フィルタ(%)	符号付整数	0 以上 100 以下
MP_HEAD_COMP	頭部拡張補正(%)	符号付整数	0 以上 100 以下
MP_BLOCK_COH	ブロック結合度	符号付整数	1 以上 10 以下または 2147483647 (0x7FFFFFFF)

* Msako 停止時のみパラメータの設定可能。パラメータの取得は常時可能。

byte_value

バイトの値

word_value

ワードの値

uint_value

符号なし整数の値

int_value

符号付整数の値

long_value

倍精度整数の値

float_value

単精度浮動小数の値

double_value

倍精度浮動小数の値

パラメータ設定メッセージ

このメッセージは、ユーザのアプリケーションプログラムが Msako のパラメータの値を設定するためのメッセージである。Msako は、このメッセージを受信すると、指定されたパラメータの値を変更する。このメッセージを送信するにあたり、ユーザのアプリケーションプログラムは送信先の Msako のウィンドウハンドルが必要となる。このために、「接続メッセージ」を使えば、あらかじめ Msako のウィンドウハンドルを入手することができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_COPYDATA	パラメータを設定する	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	User→Msako

このメッセージで使用する COPYDATASTRUCT 構造体は次のように設定する。

COPYDATASTRUCT 構造体の設定

dwData	cbData	lpData	意味
MM_SET_PARAM	MM_PARAM_DATA 構造体の大きさ	MM_PARAM_DATA 構造体へのポインタ	パラメータの値を設定する

MM_PARAM_DATA 構造体の定義

パラメータ取得メッセージの場合と同じ。

データメッセージ

このメッセージは、Msako が動体を検知したときに算出したデータをユーザのアプリケーションプログラムに知らせるためのメッセージである。ユーザのアプリケーションプログラムは「データメッセージ配信登録メッセージ」を使って、あらかじめ Msako にデータメッセージの配信を依頼しておく。Msako は、動体を検知したときに、このデータメッセージをアプリケーションプログラムへ送信する。ユーザのアプリケーションプログラムは、受信したデータメッセージから得たデータを使って独自の処理を行うことができる。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_COPYDATA	動体検知のデータ	送信元ウィンドウハンドル	COPYDATASTRUCT 構造体へのポインタ	Msako→User

このメッセージで使用する COPYDATASTRUCT 構造体は次のように設定される。

COPYDATASTRUCT 構造体の設定

dwData	cbData	lpData	意味
MM_MOTION	MM_MOTION_DATA 構造体の大きさ	MM_MOTION_DATA 構造体へのポインタ	動体検知のデータ

MM_MOTION_DATA 構造体の定義

```
typedef struct
{
    SYSTEMTIME timestamp;
    SYSTEMTIME eventstart;
    int eventno;
    int frameno;
    int volume;
    int numtarget;
    TARGET targets[MAX_TARGET];
}
MM_MOTION_DATA, *PMM_MOTION_DATA;
```

timestamp

検知したフレームの日時。

eventstart

イベントが開始した日時。

eventno

イベント番号。

frameno

フレーム番号。

volume

総移動ピクセル数。

numtarget

認識されたターゲット数。

targets

ターゲットごとの情報。TARGET 構造体の配列。配列の大きさ MAX_TARGET は 3。配列の先頭から numtarget 個分のターゲットの情報が埋められている。

TARGET 構造体の定義

```
typedef struct
{
    int label;
    int weight;
    int center_x;
    int center_y;
    int min_x;
    int max_x;
    int min_y;
    int max_y;
    COLORREF color;
}
TARGET, *PTARGET;
```

label

ターゲットの識別子。ただし、フレーム間での一意性はない。

weight

ターゲットに含まれる移動ピクセル数。

center_x

center_y

ターゲットの重心の座標。

min_x

max_x

min_y

max_y

ターゲットの範囲。

color

ターゲットの色。

バージョン取得メッセージ

このメッセージは、Msako のバージョンを知るために使う。まず、ユーザのアプリケーションプログラムは要求のメッセージを送信する。このメッセージを受取った Msako は、要求元のユーザのアプリケーションプログラムへ応答のメッセージを送信する。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_VERSION	バージョン取得 (要求)	送信元ウィンドウハンドル	NULL	User→Msako
WM_MSAKO_VERSION	バージョン取得 (応答)	送信元ウィンドウハンドル	MM_VERSION 構造体	Msako→User

MM_VERSION 構造体の定義

```
typedef union
{
    UINT uint_value;
    struct
    {
        BYTE major;
        BYTE minor;
        BYTE micro;
        BYTE revision;
    }
    byte_values;
}
MM_VERSION, *PMM_VERSION;
```

uint_value

バージョンの整数値。

major

メジャーバージョン。

minor

マイナーバージョン。

micro

マイクロバージョン。

revision

リビジョン。

再構成メッセージ

このメッセージは、Msako に対しその設定値を再度読み込むこと指示する。まず、ユーザのアプリケーションプログラムは要求のメッセージを送信する。このメッセージを受取った Msako は、要求元のユーザのアプリケーションプログラムへ応答のメッセージを送信する。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_CONFIGURE	再構成せよ(要求)	送信元ウィンドウハンドル	NULL	User→Msako
WM_MSAKO_CONFIGURED	再構成結果(応答)	送信元ウィンドウハンドル	BOOL 値	Msako→User

BOOL 値 = 真:成功/偽:失敗

再起動メッセージ

このメッセージは、Msako に対し再起動すること指示する。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_MSAKO_RESET	再起動せよ	送信元ウィンドウハンドル	NULL	User→Msako

外部機器警報メッセージ

このメッセージは、各種センサが捉えた異常を Msako に通知するために使用する。

メッセージ ID	意味	WPARAM	LPARAM	方向
WM_SAKO_ALARM	警報	送信元ウィンドウハンドル	MM_ALARM 構造体	User→Msako

MM_ALARM 構造体の定義

```
typedef union
{
    UINT uint_value;
    struct
    {
        BYTE reserved[3];
        BYTE channels;
    };
    struct
    {
        UINT reserved_bits : 24;
        UINT ch8 : 1;
        UINT ch7 : 1;
        UINT ch6 : 1;
        UINT ch5 : 1;
        UINT ch4 : 1;
        UINT ch3 : 1;
        UINT ch2 : 1;
        UINT ch1 : 1;
    };
}
MM_ALARM, *PMM_ALARM;
```

uint_value

警報データの整数値。

channels

警報データのバイト値。

ch1 ~ ch8

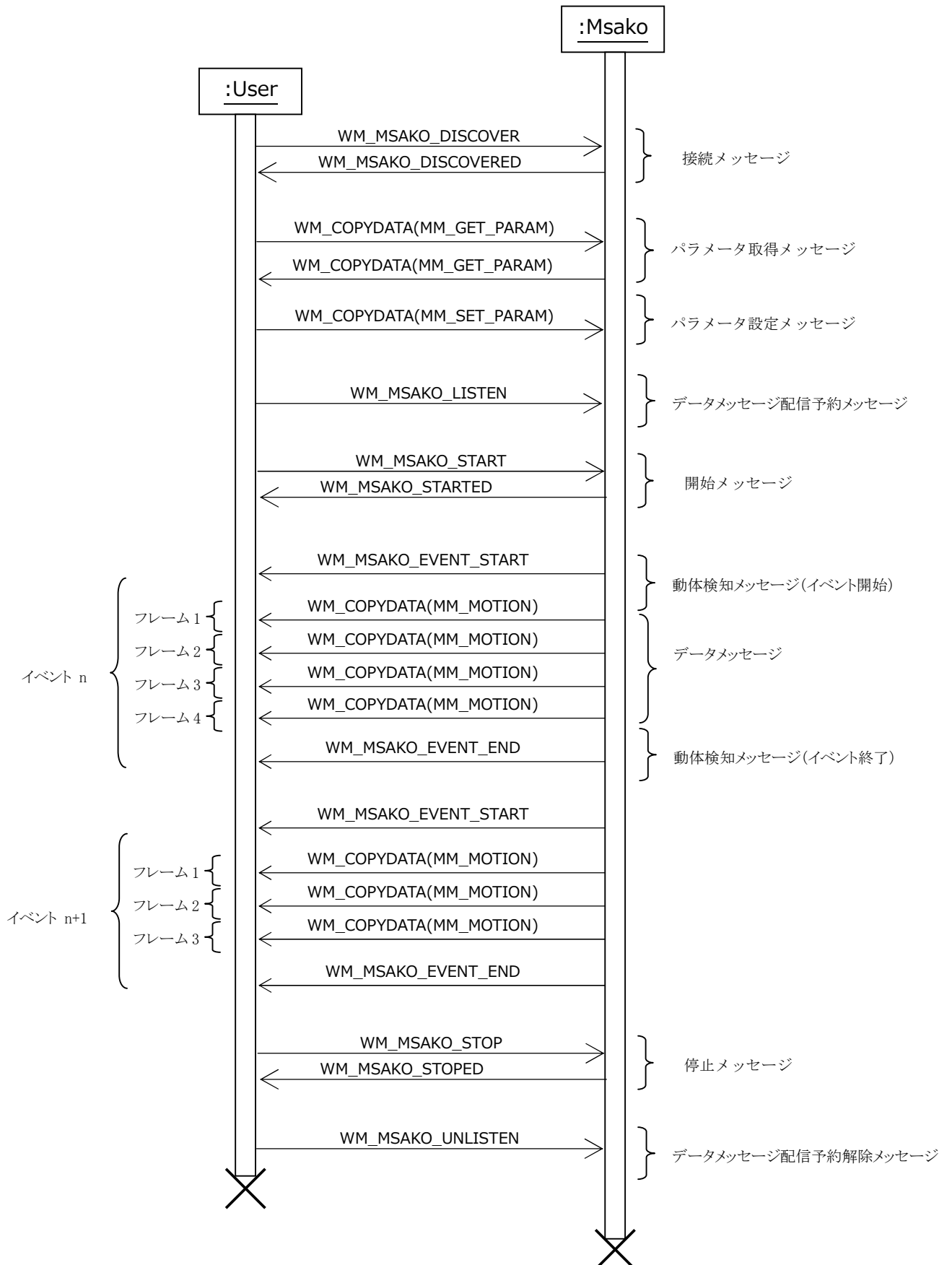
警報データのビット値。どのチャンネルで警報が発生したかを示す。

reserved

reserved_bits

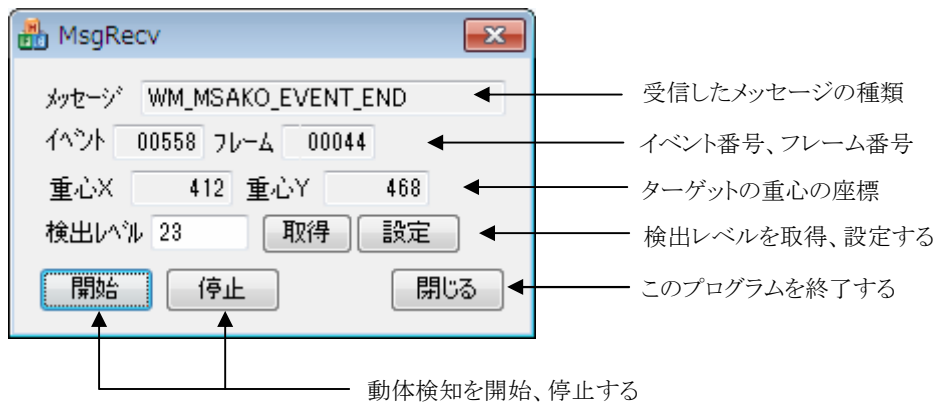
予約。

典型的なメッセージ通信手順



サンプルプログラム

このサンプルプログラムは、メッセージを使って Msako と通信する方法をデモンストレーションする。このプログラムは、VisualStudio 2008 Professional Edition を使って作成された。このプログラムを起動すると、下のよう画面が表示される。



Msako のコマンド実行タブでは、Msako が送信するメッセージを選択することができます。

