
# 宇宙機での ROSアプリケーション活用

**斎藤達彦**[1]

加藤裕基[1], 平野大地[1], 岩渕甲誠[2], 川口仁[2]

[1] 宇宙航空研究開発機構 (JAXA), [2] 株式会社セック

# 目次

- イントロダクション
  - 宇宙ロボットソフトウェア
  - 宇宙機用ソフトウェア（CFS）
- ROSアプリケーション移植環境
- 動作検証
- まとめ
- 今後の展望

ROS

×

cFS

(flight software framework)

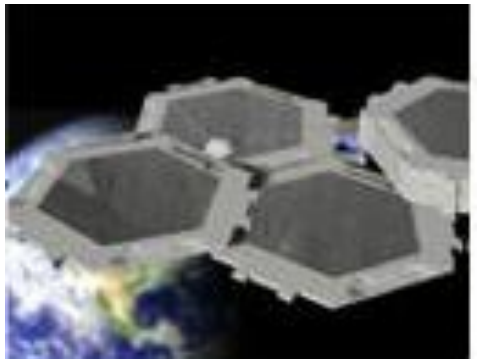
**あなたの宇宙ロボット開発参入を実現！**

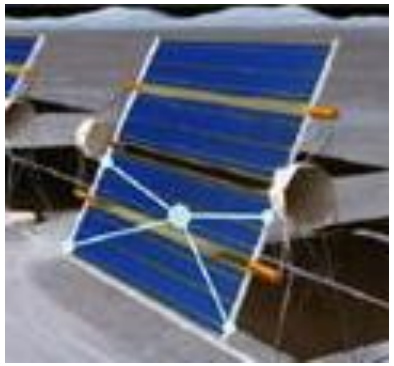
# ロボットを用いた宇宙ミッション

今後の宇宙ミッション

- 大型宇宙構造物建設（宇宙太陽光発電 等）
- 人工衛星の保守・修理
- 月・惑星探査
- スペースデブリ除去

➡ 自律ロボットの大規模ソフトウェアを駆使



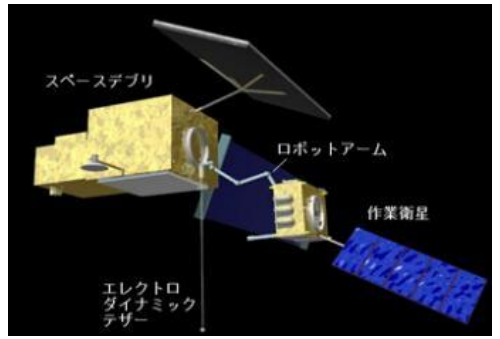


➡ **宇宙ロボットソフトウェアプラットフォーム**が必要

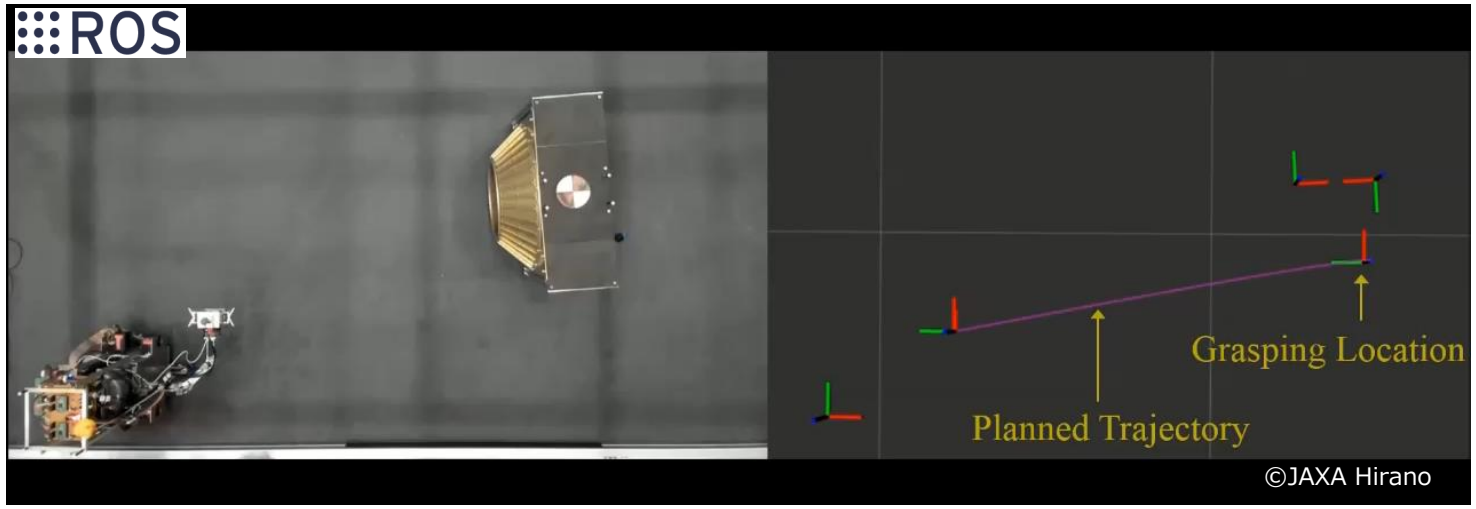
✓既存のロボット技術を容易に統合可能

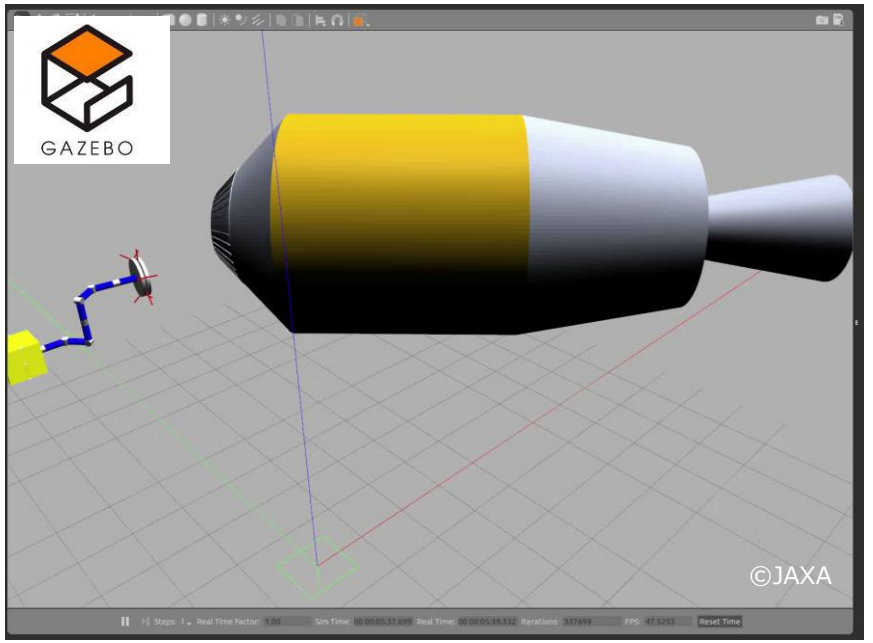
✓アプリケーション開発が容易

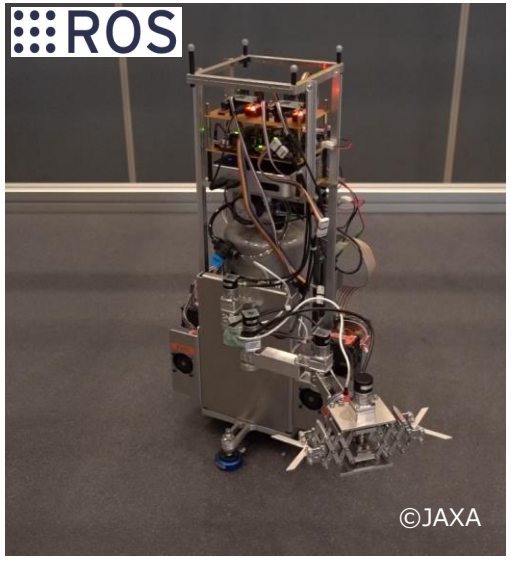
➡ **ROSエコシステムを活用**

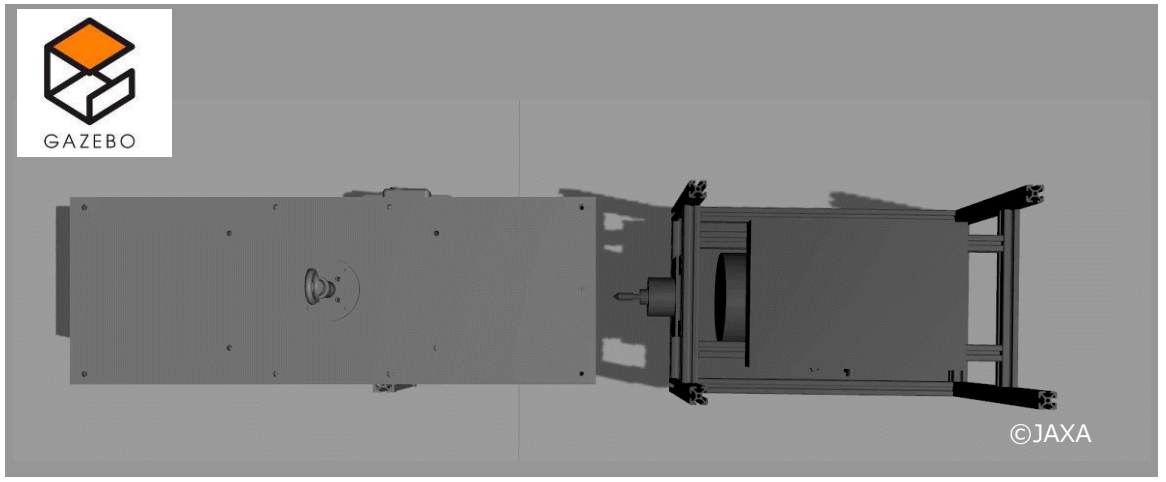


# ROSを利用した宇宙ロボティクス研究



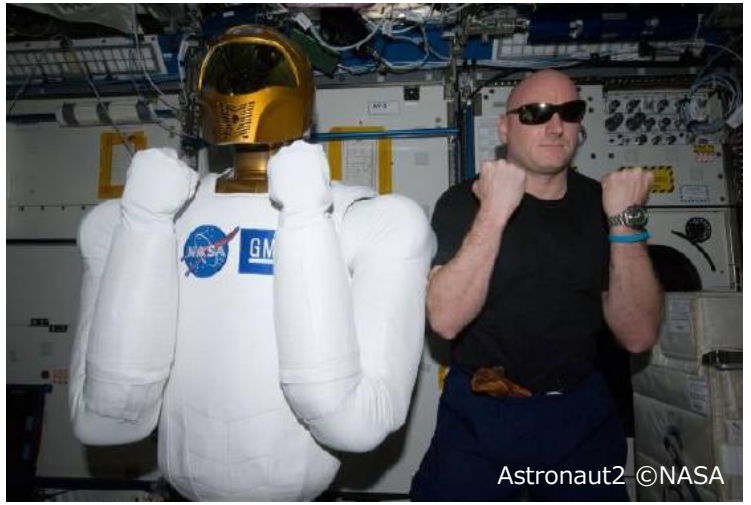




- 空気浮上ロボット制御
- 接触シミュレータ
- 捕獲シミュレータ

# 宇宙でのROS利用状況

- ISS船内ロボット
  - Robonaut2
  - Astrobee (2019年打ち上げ予定)
- ISS船外での利用実績なし
- 宇宙機利用
  - ➡ ROSを宇宙機組込み仕様に改修必要
  - ➡ コスト大


Astronaut2 ©NASA

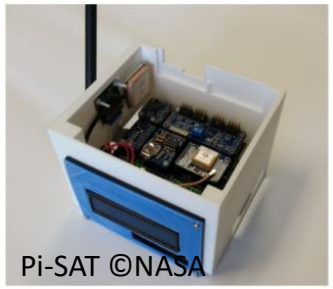
Astrobee ©NASA

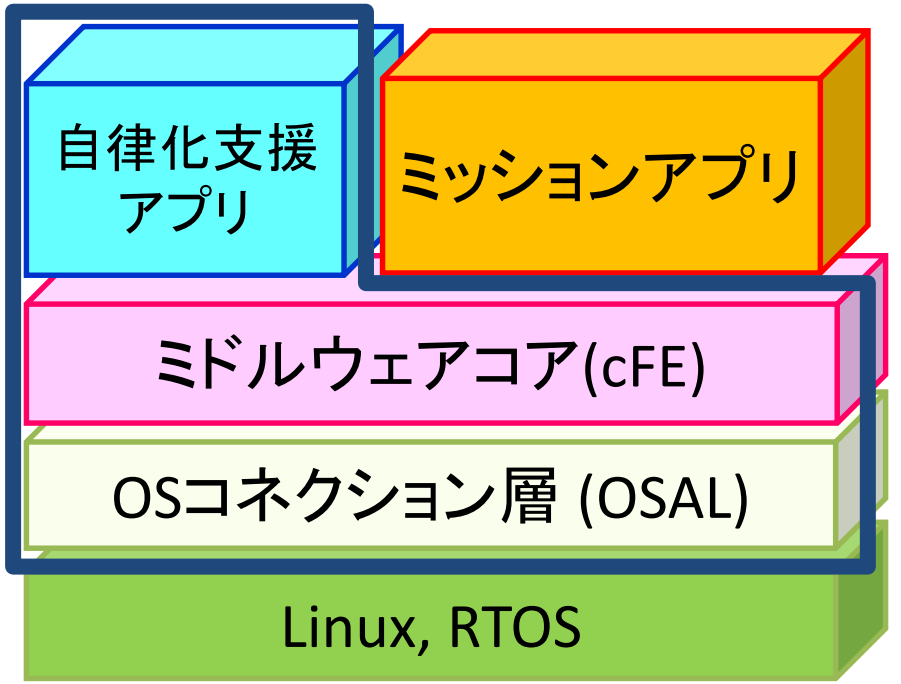
**既存の宇宙機システムにROSを組み込めないか？**

# Core Flight System(CFS)

- CFS: オープンソースの宇宙機制御ソフトウェア （©GSFC/NASA）
  - OSコネクション層（Linux, RTOS対応）
  - ミドルウェアコア（タスク・データ管理、通信制御、時刻管理、等）
  - 自律化支援アプリケーション（ログチェック、アプリ生存監視、等）
- 豊富な搭載実績：高信頼性



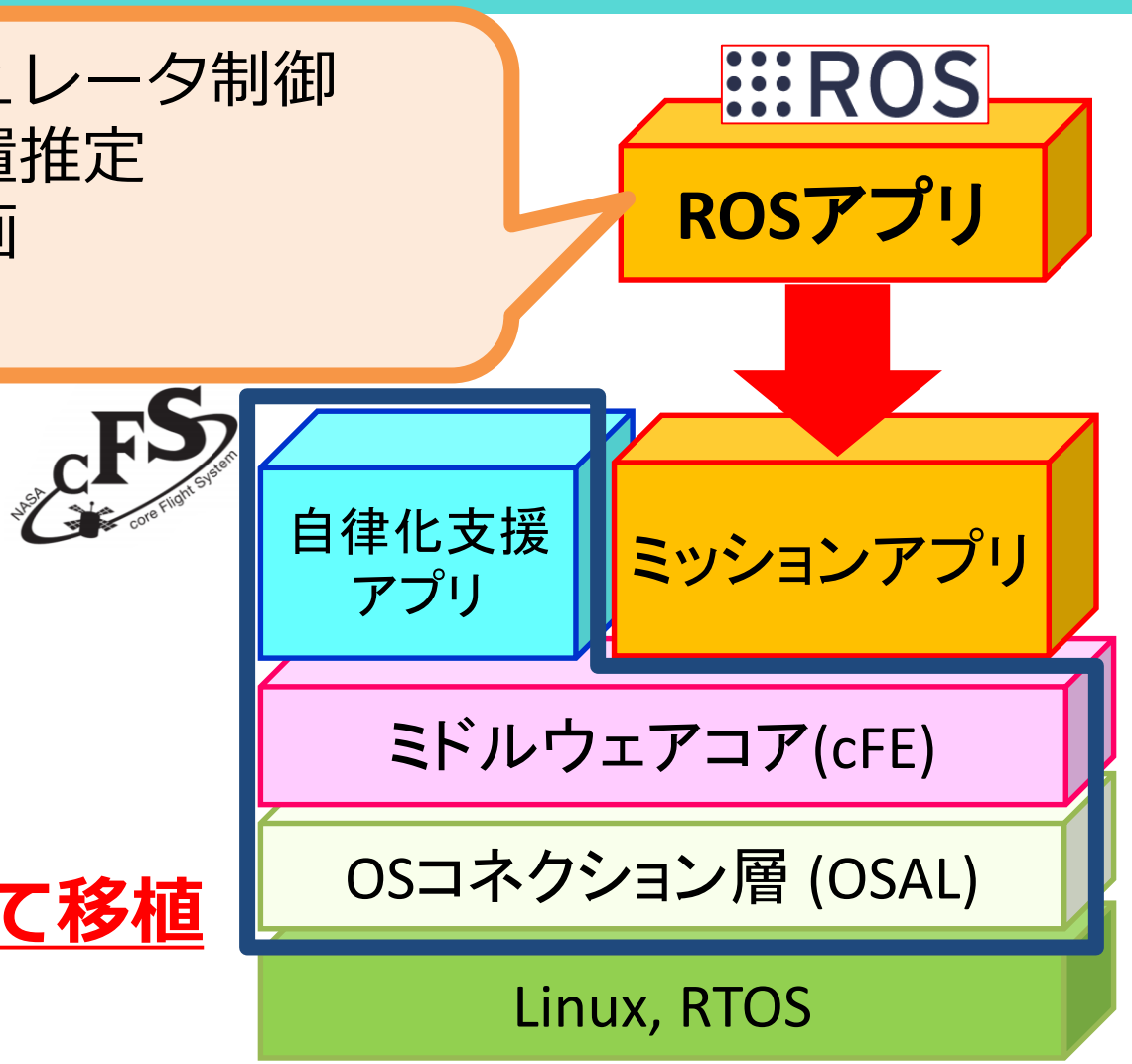



# CFSへのROS適用方法

- CFSではミッションアプリは独自開発が必要

- 我々のアプローチ：

**既存のロボットで利用されている**
**ROSアプリ（ノード）を**
**宇宙機（CFS）のミッションアプリとして移植する**



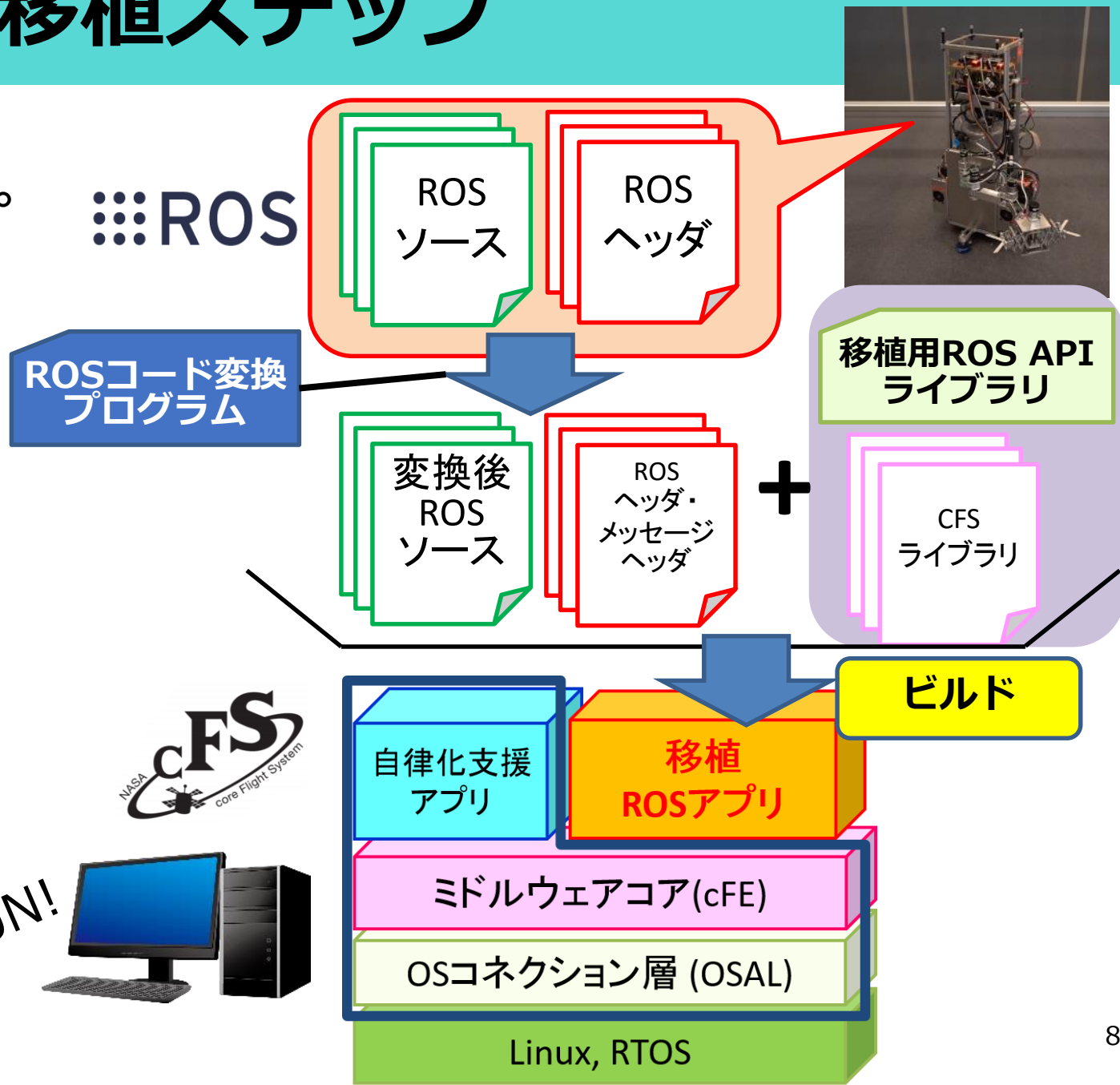
# ROSアプリケーション移植ステップ

① 移植したいROSコードを用意。
（サポート・互換性を確認）

② **ROSコード変換プログラム**を実行。コードを自動変換。

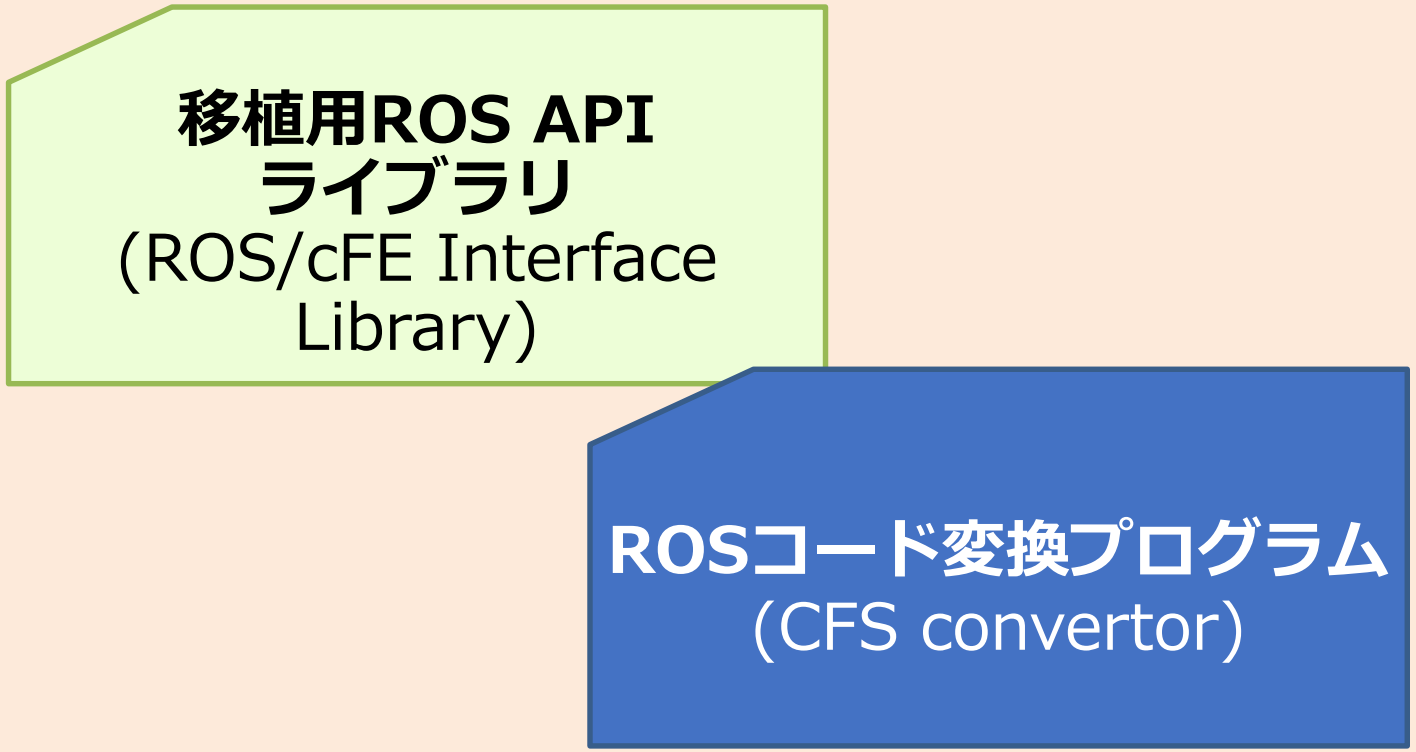
③ 自動変換されたコードを**移植用ROS APIライブラリ**、CFSコードと共にビルド。

④ 実行。
（Linux上でも動作可能）



# ROSアプリケーションの移植環境開発

- 開発コンセプト：ROSアプリのコードをほぼそのまま利用する。
  （移植の際のユーザのコストを最小化する）




【動作環境】
- ROS: Kinetic Kame
- ROS client: roscpp
  （C++コードアプリ）

# 移植用ROS API ライブラリ

**移植用ROS API ライブラリ**
(ROS/cFE Interface Library)

- 移植ROSアプリケーションが利用（ビルド時に追加する）
- CFSでのROS関数の機能を提供
- 現在は
  - ✓ **ノード間のトピック通信**
  - ✓ **周期動作（コールバック）**

  に必要な機能をサポート

利用可能なROS関数一覧

| クラス | 関数名 |
|---|---|
| - | init |
| Time | now().toSec() |
| - | ok() |
| Rate | sleep() |
| Duration | sleep() |
| - | spinOnce() |
| - | spin() |
| NodeHandle | subscribe |
| | advertise |
| Publisher | publish |

# ROSコード変換プログラム

**ROSコード変換プログラム**
(CFS convertor)

- ROSコードをCFSとビルド可能な形式に自動変換
- ROSメッセージ定義ファイルを自動付加
- ROSパッケージヘッダを自動付加

利用可能なROSメッセージ一覧

| メッセージクラス | メッセージタイプ |
|---|---|
| std msgs | header |
| | Float64MultiArray |
| | MultiArrayLayout |
| | MultiArrayDimension |
| geometry msgs | Wrench / WrenchStamped |
| | Pose / PoseStamped |
| | PoseWithCovarianceStamped |
| | Point |
| | Twist / TwistWithCovariance |
| | Vector3 |
| nav msgs | Odometry |
| sensor msgs | JointState |

# ROSコード変換プログラム

**ROSコード変換プログラム**
(CFS convertor)

- ROSコードをCFSとビルド可能な形式に自動変換
- ROSメッセージ定義ファイルを自動付加
- ROSパッケージヘッダを自動付加

利用可能なROSパッケージヘッダファイル

| パッケージ | ヘッダファイル | クラス |
|---|---|---|
| tf | Matrix3x3.h | Matrix3x3 |
| | MinMax.h | - |
| | QuadWord.h | QuadWord |
| | Quaternion.h | Quaternion |
| | Scalar.h | - |
| | StampedTransform.h | StampedTransform |
| | Transform.h | Transform |
| | Vector3.h | Vector3 |

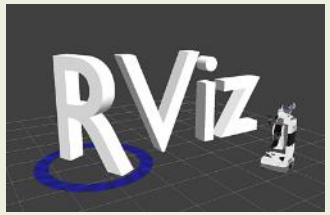
現在はTFパッケージの一部のみサポート
（今後サポート拡大予定）

# ROSコード変換プログラム

```cpp
#include "sample_pub/sample_pub.h"
int main(int argc, char** argv)
{
    ros::init(argc, argv, "sample_pub");
    ros::NodeHandle nh;
    ros::Publisher ros_pub;
    ros_pub = nh.advertise<std_msgs::Header>("ros_cfe_msg", 100);
    ros::Rate loop_rate(100);
    int count = 0;
    while(ros::ok())
    {
        std_msgs::Header msg;
        msg.seq = count;
        ROS_INFO("send msg.data: %d, rostimenow.toSec: %f", msg.seq,
                ros::Time::now().toSec());
        ros_pub.publish(msg);
        loop_rate.sleep();
        count++;
    }
    return 0;
}

```

```cpp
#include "sample_pub/sample_pub.h"
extern "C"{
    #include "cfe.h"
}
#include "convert_lib.h"
#define EVT_COUNT 1
#define EVT_INFO0_EID 1
CFE_EVS_BinFilter_t EventFilterssample_pub_1[EVT_COUNT];
int topicNo_sample_pub_1_0;
void main_sample_pub_1_(){
    EventFilterssample_pub_1[0].EventID = EVT_INFO0_EID;
    EventFilterssample_pub_1[0].Mask = CFE_EVS_NO_FILTER;
    CONVERT_RosInit(EVT_COUNT, EventFilterssample_pub_1);


    topicNo_sample_pub_1_0 = CONVERT_RosNodeHandleAdvertise("0x0000000000008", 0x1900, 100);

    uint32 loop_rate = CONVERT_RosRate(100);
    int count = 0;
    while(CONVERT_RosOk())
    {
        std_msgs::Header msg;
        msg.seq = count;
        CFE_EVS_SendEvent(EventFilterssample_pub_1[0].EventID, CFE_EVS_INFORMATION,
                          "send msg.data: %d, rostimenow.toSec: %f", msg.seq, CONVERT_RosTimeNowToSec());
        msg.vector2pointer();
        CONVERT_RosPublisherPublish((CFE_SB_Msg_t*)(&msg), sizeof(msg),topicNo_sample_pub_1_0);
        CONVERT_RosRateSleep(loop_rate, topicNo_sample_pub_1_0);
        count++;
    }

}

```

# ROSアプリケーション移植ステップ



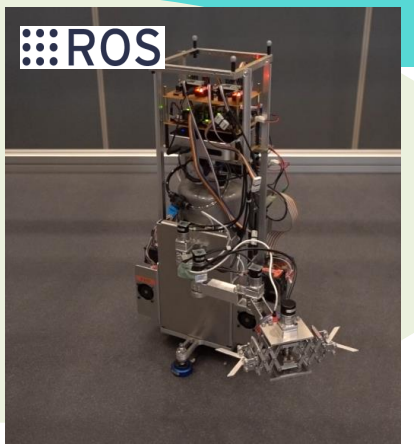

新規
ROSアプリ


ROS

ROS
ソース

ROS
ヘッダ

既存
ROSアプリ




ROSコード変換
プログラム

変換後
ROS
ソース

ROS
ヘッダ・
メッセージ
ヘッダ

+

移植用ROS API
ライブラリ

CFS
ライブラリ


自律化支援
アプリ

移植
ROSアプリ

ビルド

ミドルウェアコア(cFE)

OSコネクション層 (OSAL)

Linux, RTOS


**宇宙ロボットアプリ
開発が容易に実施可能**

# まとめ

- ROSアプリケーションの
  宇宙機システム（CFS）上での
  移植・動作環境を構築

- ROS開発ツール、ライブラリを利用
  してミッションアプリケーション
  開発効率を向上させる環境・手法
  を提案

ROS

×

cFS NASA core Flight System

**(flight software framework)**

# 今後の展望

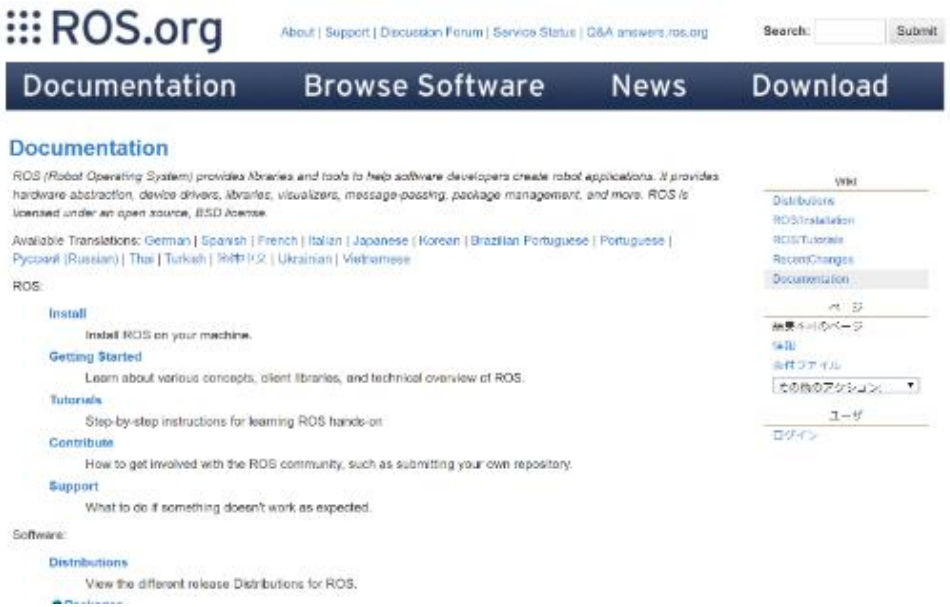
移植用ROS API ライブラリがサポートするROS機能一覧

| No | ROS 機能 | サポート状況 |
|---|---|---|
| 1. | Publish | ✓ |
| 2. | Subscribe | ✓ |
| 3. | Service | (将来対応) |
| 4. | Param | (将来対応) |
| 5. | Bag | (将来対応) |

- 現時点で未サポートのrosservice, rosparam, rosbagについて対応。
- サポート可能なメッセージクラス、ROSパッケージの拡充。
- ROS、CFSのネットワークブリッジ機能開発。
- フライト検証を計画中。

# 今後の展望

本開発成果は、今後オープンソースプロジェクトとして公開予定。





**ROSユーザの皆様のご意見・ご要望をお待ちしております**