データの読み方 その２

AMTK（AMSR2 I/O Toolkit）
利用編

2013年2月1日

RESTEC 小林和史

講義の流れ

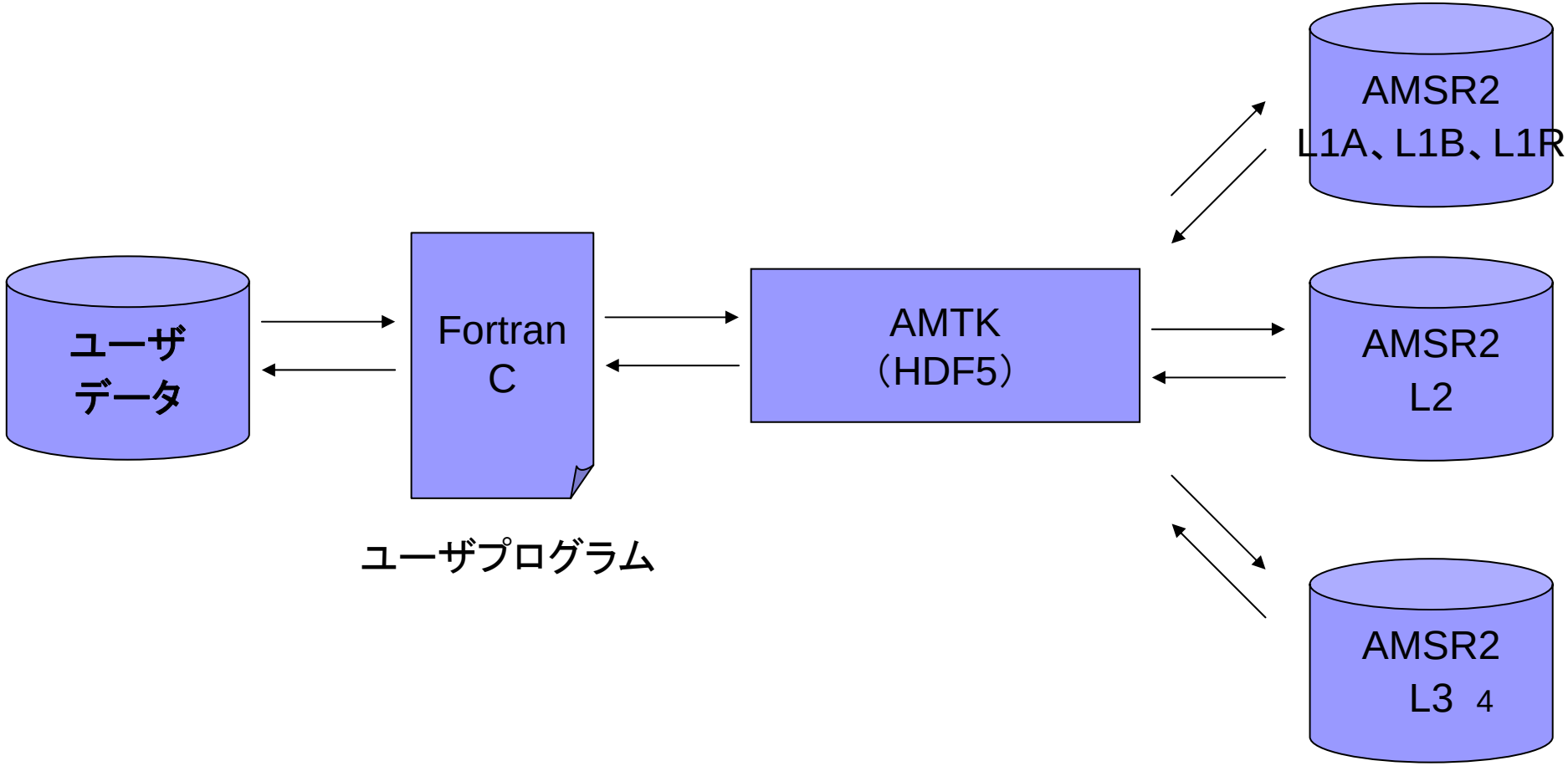
- AMTKの説明
  - AMTKとは
  - AMTKの特徴
  - AMTKの動作環境および取得先
- AMTKを用いたAMSR2データの読み方
  - HDF5について
  - 関数使用例
  - コンパイル・プログラム実行例

AMTKとは

- AMTK（AMSR2 プロダクト I/O Toolkit）とは…
  - AMTKはHDF5ライブラリをベースとしてAMSR2データ専用に開発されたツールキット
  - AMSR2データはHDF5のファイル形式で格納されており、C言語やFortran言語のプログラムで利用するためには、HDF5ライブラリについての理解が必要
  - AMTKを利用することでHDF5形式のファイルが利用しやすくなる

AMTKの特徴

- AMSR2データが容易に読み込み可能
- 任意のデータをAMSR2形式に出力可能

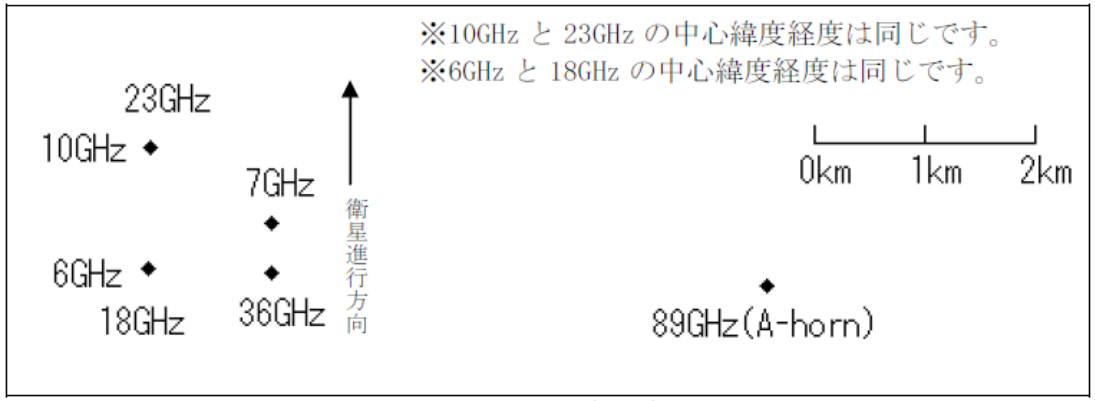


AMTKの特徴

- 時刻の変換
  - TAI93：1993 年1 月1 日0 時を起点とした通算秒
  - TAI93を年/月/日/時/分/秒形式に変換してくれる
- 格納データのスケール処理
  - ファイル容量節約のため、輝度温度などのデータは整数値で格納されている
  - 例：輝度温度（273.15K）は27315という値で格納されている
    →スケール0.01を自動的に掛けてくれる
- 異常データの判定
  - 上記スケール処理の際、異常データを自動で判定し処理は行わない
  - 例：欠損データ（65535）→ 655.35にはならない

AMTKの特徴

- L1低周波チャンネルの緯度経度を自動的に算出
  - L1プロダクトには89GHzチャンネルの緯度経度しか格納されていない
  - AMSR2は周波数チャンネルごとに観測している緯度経度が異なるため、低周波チャンネルの緯度経度はユーザー独自で求める必要がある



AMTKの動作環境

■ 動作確認環境

機種名	DELL4		Sun-Fire-V440	SGI Origin2000
OS	Red Hat Enterprise Linux Client release 5 (Tikanga)		SunOS 5.9	IRIX64 6.5
コンパイラ	C 言語	Fortran	Sun Studio 11	MIPSpro Compilers: Version 7.30
	gcc, g++ Intel C++ Compiler	g77, g95 (※1) f77, f95 Intel Fortran Compiler PGI Fortran Compiler		
メモリサイズ	12GB		8GB	1GB
HDF ライブラリ	HDF5-1.8.4-patch1 (※2)			

(※1)g95 コンパイラは、G95 の Web サイトにて公開されている以下のバイナリを使用しました。

Linux x86_64/EMT64 (32 bit D.I.) Default integer of 32 bits, compatible with older programs

(※2)

Sun-Fire-V440 は HDF ライブラリをソースコードからコンパイルする必要があります。

SGI Origin2000 は HDF ライブラリのソースコードを一部改変してコンパイルする必要があります。

AMTKの取得先

- HDF5ライブラリ
- SZIPライブラリ
  - The HDF Groupのホームページ
    - http://www.hdfgroup.org/HDF5/
- AMTK
  - GCOM-W1データ提供サービスホームページ
    - http://gcom-w1.jaxa.jp/
  - GCOM-W1 ホームページ
    - http://suzaku.eorc.jaxa.jp/GCOM_W/index_j.html

AMTKを用いたAMSR2データの読み方

- AMSR2プロダクト（HDF5）について
  - ヘッダ部とデータ部が存在する
    - ヘッダ部：プロダクトメタデータ（プロダクト固有情報）を格納
    - データ部：観測データなどn次元の格子状データを格納

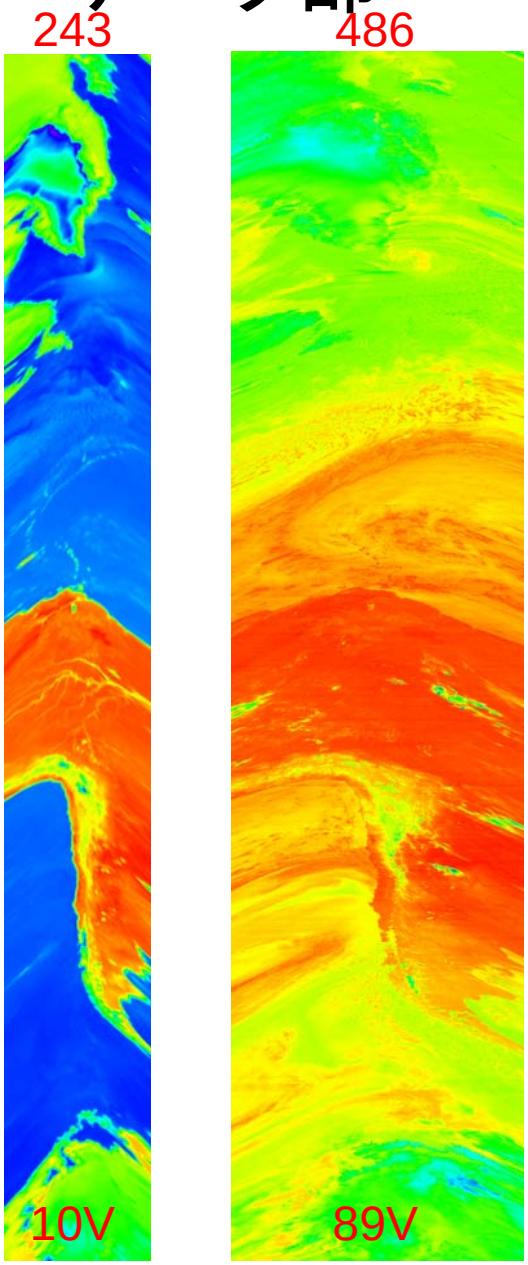
ヘッダ部

AMSRBandWidth	6G-350MHz_7G-350MHz,10G-100MHz,18G-200MHz,23G-400MHz,36G-1000MHz,89GA-3000MHz,89GB
AMSRBeamWidth	6G-1.8deg,7G-1.8deg,10G-1.2deg,18G-0.65deg,23G-0.75deg,36G-0.35deg,89GA-0.15deg,89GB-0.15deg
AMSRChannel	6.925GHz,7.3GHz,10.65GHz,18.7GHz,23.8GHz,36.5GHz,89.0GHz-A,89.0GHz-B
AlgorithmVersion	0
AncillaryDataInformation	
AntennaRotationVelocity	40
AttitudeMissingDataRate	Good
AutomaticQAFlag	Good
AutomaticQAFlagExplanation	1:MissingScanQA:Less than 21 is available->OK,2:MissingDataQA:Less than 321 is available->OK,3:Ante
CSMTemperature	6GV-2.700,6GH-2.700,7GV-2.700,7GH-2.700,10GV-2.700,10GH-2.700,18GV-2.800,18GH-2.800,23GV-2.7
CalibrationCurveCoefficient#1	6GV-0.000000,6GH-0.000000,7GV-0.000000,7GH-0.000000,10GV-0.000000,10GH-0.000000,18GV-0.000
CalibrationCurveCoefficient#2	6GV-1.000000,6GH-1.000000,7GV-1.000000,7GH-1.000000,10GV-1.000000,10GH-1.000000,18GV-1.000
CalibrationCurveCoefficient#3	6GV-0.000000,6GH-0.000000,7GV-0.000000,7GH-0.000000,10GV-0.000000,10GH-0.000000,18GV-0.000
CalibrationCurveCoefficient#4	6GV-0.000000,6GH-0.000000,7GV-0.000000,7GH-0.000000,10GV-0.000000,10GH-0.000000,18GV-0.000
CalibrationCurveCoefficient#5	6GV-0.000000,6GH-0.000000,7GV-0.000000,7GH-0.000000,10GV-0.000000,10GH-0.000000,18GV-0.000
CalibrationMethod	RxTemperatureReferencedSoilOverCSMInterpolation,Absolute89GPositioningNonlinearityCorrection
CoRegistrationParameterA1	6G-1.25000,7G-1.00000,10G-1.25000,18G-1.25000,23G-1.25000,36G-1.00000
CoRegistrationParameterA2	6G-0.00000,7G--0.10000,10G--0.25000,18G-0.00000,23G--0.25000,36G-0.00000
CoefficientAhh	6G--1.027,7G--1.037,10G--1.023,18G--1.017,23G--1.026,36G--1.021,89GA--1.020,89GB--1.019
CoefficientAhv	6G---0.003,7G---0.009,10G---0.003,18G---0.003,23G---0.003,36G---0.002,89GA---0.003,89GB---0.002
CoefficientAoh	6G---0.066,7G---0.078,10G---0.054,18G---0.039,23G---0.065,36G---0.051,89GA---0.058,89GB---0.054
CoefficientAov	6G---0.066,7G---0.078,10G---0.054,18G---0.039,23G---0.065,36G---0.051,89GA---0.058,89GB---0.054
CoefficientAvh	6G---0.003,7G---0.008,10G---0.003,18G---0.003,23G---0.003,36G---0.002,89GA---0.002,89GB---0.002
CoefficientAvv	6G-1.027,7G-1.038,10G-1.023,18G-1.017,23G-1.026,36G-1.021,89GA-1.020,89GB-1.019
ContactOrganizationName	JAXA GCOM Project
ContactOrganizationTelephone	
DataFormatType	HDF
DynamicRange	2.7K-340K
EGSDataModel	B.0
EllipsoidName	WGS84
EphemerisMissingDataRate	Good
EphemerisQA	OK
EquatorCrossingDateTime	2012-08-04T12:55:50.843Z
EquatorCrossingLongitude	-172.17
FlatteningRatioofEarth	0.00335
GeophysicalName	Brightness Temperature
GlobalMeteorologicalDataType	
GranuleID	GW1AM2_201208041229_070D_L1SGBTBR_0000000
GringPointLatitude	84.27,73.50,34.00,-24.92,-84.09,-73.29,-22.69,36.49
GringPointLongitude	7.39,-53.94,-154.12,-168.40,177.47,113.92,175.81,-171.60
HDFFormatVersion	Ver5.1.8.3
InputFileNames	GW1AM2_201208041141_054A_L0S1576E.bin,GW1AM2_201208041231_070D_L0S1576E.bin
NumberMissingPackets	0
NumberOfGoodPackets	31616
NumberOfInputFiles	2
NumberOfMissingScans	0
NumberOfPackets	
NumberOfScans	1976
ObservationEndDateTime	2012-08-04T13:18:25.648Z
ObservationStartDateTime	2012-08-04T12:29:03.428Z
OffNadir	47.0deg : 89GB, 47.5deg : others
Operation	Standard
OrbitArgumentPerigee	-
OrbitDataFileName	
OrbitDataType	ONBOARD
OrbitDirection	Descending
OrbitEccentricity	Frozen
OrbitInclination	98.188deg
OrbitPeriod	98.8min
OrbitSemiMajorAxis	7085.858km
OverlapScans	20
PGEName	GCOM-W1 Mission Operation System
ParameterVersion	0
PassNumber	70
PlatformShortName	GCOM-W1
Platinum1ConversionTableW0	-35.000000,-48.030000,75.000000
Platinum1ConversionTableW1	0.000000,0.033900,0.000000
Platinum1ConversionTableW2	0.000000,0.000000,0.000000
Platinum1ConversionTableW3	0.000000,0.000000,0.000000
Platinum1ConversionTableW4	0.000000,0.000000,0.000000
Platinum1CountRange	0,385,3630,4096
Platinum2ConversionTableW0	-150.000000,-163.430000,150.000000
Platinum2ConversionTableW1	0.000000,0.090900,0.000000
Platinum2ConversionTableW2	0.000000,0.000000,0.000000
Platinum2ConversionTableW3	0.000000,0.000000,0.000000
Platinum2ConversionTableW4	0.000000,0.000000,0.000000
Platinum2CountRange	0,148,3452,4096
Platinum3ConversionTableW0	-242.103000,-241.644000,-241.684000,-241.657000,-241.678000,-241.460000,-241.497000,-241.722000,
Platinum3ConversionTableW1	0.116812,0.116174,0.116131,0.116165,0.116180,0.115852,0.115839,0.116029,0.115861,0.115949
Platinum3ConversionTableW2	0.000002328650,0.000002428100,0.000002420060,0.000002422170,0.000002429980,0.000002465540,0.00
Platinum3ConversionTableW3	0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000
Platinum3ConversionTableW4	0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000
ProcessingCenter	JAXA GCOM Project
ProcessingQAAttribute	
ProcessingQADescription	PROC COMP
ProductName	AMSR2-L1B
ProductSize_MByte	49.4
ProductVersion	0
ProductionDateTime	2012-11-12T09:05:01.000Z
QALocationOfPacketDiscontinuity	Continuation
QAPercentMissingData	0
QAPercentOutofBoundsData	0
QAPercentParityErrorData	0
RevisitTime	16days
SatelliteAltitude	699.6km
SatelliteOrbit	Sun-synchronous_sub-recurrent
ScanningPeriod	1.5sec
ScienceQualityFlag	
ScienceQualityFlagExplanation	-
SemiMajorAxisofEarth	6378.1km
SensorAlignment	Rx
SensorShortName	AMSR2
SpatialResolution	6G-35kmX61km,7G-35kmX61km,10G-24kmX41km,18G-13kmX22km,23G-15kmX26km,36G-7kmX12km,8
StartOrbitNumber	1150
StopOrbitNumber	1150
SwathWidth	1450km
Thermistor1ConversionTableD	0.000000,3.333300,2.439800,0.000000
Thermistor1ConversionTableE	0.000000,1038.960000,1064.250000,0.000000
Thermistor1ConversionTableF	-35.000000,-42.222200,-28.363100,80.000000
Thermistor1CountRange	0,1044,1201,3027,4096
Thermistor2ConversionTableW0	50.000000,54.797000,-25.000000
Thermistor2ConversionTableW1	0.000000,-0.03346800,0.00000000
Thermistor2ConversionTableW2	0.000000000,0.000008167,0.000000000
Thermistor2ConversionTableW3	0.000000000000000,-0.000000001303700,0.000000000000000
Thermistor2ConversionTableW4	0.000000,0.000000,0.000000
Thermistor2CountRange	0,149,3899,4096
Thermistor3ConversionTableW0	65.000000,146.740000,-35.000000
Thermistor3ConversionTableW1	0.000000,-0.164500,0.000000
Thermistor3ConversionTableW2	0.000000000,0.000083636,0.000000000
Thermistor3ConversionTableW3	0.000000000000,-0.000000020322,0.000000000000
Thermistor3ConversionTableW4	0.000000000000000,0.000000000018518,0.000000000000000
Thermistor3CountRange	0,714,3431,4096

- AMSR2 L1Bプロダクトのヘッダ部には122個のメタデータが格納されている

メタデータ名	格納メタデータ
GeophysicalName	Brightness Temperature
NumberOfScans	1976
ObservationEndDateTime	2012-08-04T13:18:25.648Z
ObservationStartDateTime	2012-08-04T12:29:03.428Z

データ部



- AMSR2には低解像度（低周波チャンネル）のデータと高解像度（高周波チャンネル）のデータが存在する
- AMSR2データ部の観測幅のサンプル数（ピクセル数）は低解像度が243点、高解像度が486点
- AMSR2データ部の観測スキャン数はプロダクトにより異なる

AMTKの関数

- AMTKにはAMSR2プロダクトを簡易的に読み込むための関数が存在する（Fortran・C共通の関数名）
  - HDFファイルオープン関数：**AMTK_openH5**
  - HDFファイルクローズ関数：**AMTK_closeH5**
  - メタデータ取得関数：**AMTK_getMetaDataName**
  - 時刻データ取得関数：**AMTK_getScanTime**
  - 緯度経度データ取得関数：**AMTK_getLatLon**
  - 実数型データ取得関数：**AMTK_get_SwathFloat**
  - 整数型データ取得関数：**AMTK_get_SwathInt**
  - など

- 読み込むデータによって使用する関数が異なる

AMTK関数の特徴

```
dhnd = H5Dopen(fhnd, "Scan Time", H5P_DEFAULT);
ret = H5Dread(dhnd, H5T_NATIVE_DOUBLE, H5S_ALL, H5S_ALL, H5P_DEFAULT, r8d1);
if(ret < 0){
printf("H5Dread error: Scan Time¥n");
exit(1);
}
 ret = H5Dclose(dhnd);
// get leap second file
fn = getenv(evar);
if(fn == NULL || *fn == '¥0'){
printf("amsr2time: no environment error: %s¥n", evar);
exit(1);
}
else{
printf("amsr2time: %s = %s¥n", evar, fn);
}
// open
hnd = fopen(fn, "r");
if(hnd == NULL){
printf("amsr2time: leap second file open error¥n");
exit(1);
}
// count leap second entry
lnum = 0;
while(1){
// read
ret = fgets(buf, 512, hnd);
// end check
if(ret == NULL) break;
// check & count
if(buf[0] != '/'){
sscanf(buf, "%hd", &ibuf);
if(ibuf >= 1993){
++lnum;
}}}
// read leap second data
ldat = malloc(sizeof(LEAP_SECOND) * lnum);
rewind(hnd);
i = 0;
while(1){
// read
ret = fgets(buf, 512, hnd);
// end check
if(ret == NULL) break;
// check & count
if(buf[0] != '/'){
sscanf(buf, "%hd", &ibuf);
if(ibuf >= 1993){
sscanf(buf, "%hd %hd %lf %lf %lf %lf"
, &ldat[i].year, &ldat[i].month, &rbuf, &rbuf, &rbuf
, &ldat[i].tai93sec);
printf("amsr2time: year=%4d month=%2d tai93sec=%14.2lf¥n"
, ldat[i].year, ldat[i].month, ldat[i].tai93sec);
++i;
}}}
printf("amsr2time: number of leap second = %d¥n", lnum);
// close
fclose(hnd);
// convert
for(i = 0; i < *num; ++i){
// negative value is warning
if(tai93[i] < 0){
printf("amsr2time: negative value warning: "
"%14.2lf (scan_by_1origin=%04d)¥n", tai93[i],i+1);
st[i].tai93sec = tai93[i];
st[i].year    = 0;
st[i].month   = 0;
st[i].day     = 0;
st[i].hour    = 0;
st[i].minute  = 0;
st[i].second  = 0;
st[i].ms      = 0;
st[i].reserve  = 0;
continue;
}
// check leap second & strike
lcnt = 0;
flag = 0;
for(j = lnum - 1; j >= 0; --j){
if(tai93[i] >= ldat[j].tai93sec){
lcnt = j + 1;
break;
}
else if(tai93[i] >= ldat[j].tai93sec - 1){
lcnt = j;
flag = 1;
break;
}}
// convert unix epoch time
utime = tai93[i] + 725846400 - lcnt;
// convert struct tm
stmp = gmtime(&utime);
// store result in AM2_COMMON_SCANTIME
st[i].tai93sec = tai93[i];
st[i].year    = stmp->tm_year + 1900;
st[i].month   = stmp->tm_mon + 1;
st[i].day     = stmp->tm_mday;
st[i].hour    = stmp->tm_hour;
st[i].minute  = stmp->tm_min;
st[i].second  = stmp->tm_sec;
st[i].ms      = (int)((tai93[i] - (long int)tai93[i]) * 1000 + 0.5);
st[i].reserve  = 0;
// deal for strike
if(flag){
// convert unix epoch time
utime = tai93[i] + 725846400 - lcnt - 1;
// convert struct tm
stmp = gmtime(&utime);
// store result in AM2_COMMON_SCANTIME
st[i].tai93sec = tai93[i];
st[i].year    = stmp->tm_year + 1900;
st[i].month   = stmp->tm_mon + 1;
st[i].day     = stmp->tm_mday;
st[i].hour    = stmp->tm_hour;
st[i].minute  = stmp->tm_min;
st[i].second  = 60;
st[i].ms      = (tai93[i] - (long int)tai93[i]) * 1000 + 0.5;
st[i].reserve  = 0;
}}
// free
free(ldat);
}

// read meta: CoRegistrationParameterA1
 ahnd = H5Aopen(fhnd, "CoRegistrationParameterA1", H5P_DEFAULT);
 atyp = H5Aget_type(ahnd);
 ret = H5Aread(ahnd, atyp, &buf);
 if(ret < 0){
  printf("H5Aread error: CoRegistrationParameterA1¥n");
  exit(1);
 }
 ret = H5Aclose(ahnd);
 sscanf(buf, "6G-%lf,7G-%lf,10G-%lf,18G-%lf,23G-%lf,36G-%lf"
 , &prm1[0]
 , &prm1[1]
 , &prm1[2]
 , &prm1[3]
 , &prm1[4]
 , &prm1[5]
 );
 prm1[6] = (prm1[0] + prm1[1] + prm1[2] + prm1[3] + prm1[4] + prm1[5]) / 6.0;
// read meta: CoRegistrationParameterA2
 ahnd = H5Aopen(fhnd, "CoRegistrationParameterA2", H5P_DEFAULT);
 atyp = H5Aget_type(ahnd);
 ret = H5Aread(ahnd, atyp, &buf);
 if(ret < 0){
  printf("H5Aread error: CoRegistrationParameterA2¥n");
  exit(1);
 }
 ret = H5Aclose(ahnd);
 sscanf(buf, "6G-%lf,7G-%lf,10G-%lf,18G-%lf,23G-%lf,36G-%lf"
 , &prm2[0]
 , &prm2[1]
 , &prm2[2]
 , &prm2[3]
 , &prm2[4]
 , &prm2[5]
 );
 prm2[6] = (prm2[0] + prm2[1] + prm2[2] + prm2[3] + prm2[4] + prm2[5]) / 6.0;
// read meta: CoRegistrationParameterA2
 ahnd = H5Aopen(fhnd, "CoRegistrationParameterA2", H5P_DEFAULT);
 atyp = H5Aget_type(ahnd);
 ret = H5Aread(ahnd, atyp, &buf);
 if(ret < 0){
  printf("H5Aread error: CoRegistrationParameterA2¥n");
  exit(1);
 }
 ret = H5Aclose(ahnd);
 sscanf(buf, "6G-%lf,7G-%lf,10G-%lf,18G-%lf,23G-%lf,36G-%lf"
 , &prm2[0]
 , &prm2[1]
 , &prm2[2]
 , &prm2[3]
 , &prm2[4]
 , &prm2[5]
 );
 prm2[6] = (prm2[0] + prm2[1] + prm2[2] + prm2[3] + prm2[4] + prm2[5]) / 6.0;

// parameter
pi = acos(-1.0);
rad = pi / 180.0;
deg = 180.0 / pi;

// pixel loop
for(j = 0; j < *num; ++j){
for(i = 0; i < SMPL; ++i){
// set short cut
lat1 = lathi[j][i*2+0];
lat2 = lathi[j][i*2+1];
lon1 = lonhi[j][i*2+0];
lon2 = lonhi[j][i*2+1];

// check input range
if(lat1 <  -90.0 ||  +90.0 < lat1
|| lon1 < -180.0 || +180.0 < lon1
|| lat2 <  -90.0 ||  +90.0 < lat2
|| lon2 < -180.0 || +180.0 < lon2){
printf("amsr2latlon: out of range warning: "
"latlon1(%f,%f)->latlon2(%f,%f) "
"(low_pixel_by_1origin=%d,scan_by_1origin=%d)¥n"
, lat1, lon1, lat2, lon2, i+1, j+1);
latlo[j][i] = MV;
lonlo[j][i] = MV;
continue;
}

// calculate
p1[0] = cos(lon1*rad)*cos(lat1*rad);
p1[1] = sin(lon1*rad)*cos(lat1*rad);
p1[2] = sin(lat1*rad);
p2[0] = cos(lon2*rad)*cos(lat2*rad);
p2[1] = sin(lon2*rad)*cos(lat2*rad);
p2[2] = sin(lat2*rad);
temp = p1[0]*p2[0]+p1[1]*p2[1]+p1[2]*p2[2];
theta = acos(temp);
for(k = 0; k < 3; ++k) ex[k]=p1[k];
temp = sqrt(p1[0]*p1[0]+p1[1]*p1[1]+p1[2]*p1[2])*
sqrt(p2[0]*p2[0]+p2[1]*p2[1]+p2[2]*p2[2])*sin(theta);
ez[0] = (p1[1]*p2[2]-p1[2]*p2[1])/temp;
ez[1] = (p1[2]*p2[0]-p1[0]*p2[2])/temp;
ez[2] = (p1[0]*p2[1]-p1[1]*p2[0])/temp;
ey[0] = ez[1]*ex[2]-ez[2]*ex[1];
ey[1] = ez[2]*ex[0]-ez[0]*ex[2];
ey[2] = ez[0]*ex[1]-ez[1]*ex[0];
J = cos(*prm2*theta);
K = cos(*prm1*theta);
L = sin(*prm1*theta);
M = sin(*prm2*theta);
pt[0] = J*(K*ex[0]+L*ey[0])+M*ez[0];
pt[1] = J*(K*ex[1]+L*ey[1])+M*ez[1];
pt[2] = J*(K*ex[2]+L*ey[2])+M*ez[2];
temp = sqrt(pt[0]*pt[0] + pt[1]*pt[1]);
lonlo[j][i] = atan2(pt[1],pt[0])*deg;
latlo[j][i] = atan2(pt[2],temp )*deg;
}
}
}
```

- 時刻と緯度経度の読み込み
- AMTKを使用しない場合、プログラムは左のように膨大
- AMTKを使用すると下の2行で同じデータを読み込める

ret=AMTK_getScanTime(hnd,1,num,st)
ret=AMTK_getLatLon(hnd,LL06,1,num,AM2_LATLON_06)

AMSR2データの読み込みの流れ

- AMSR2 L1Bプロダクトのヘッダ部とデータ部を読み込んで、内容を表示する
  - HDFファイルのオープン
  - ヘッダ部の読み込み
  - データ部の読み込み
  - HDFファイルのクローズ
- Fortran言語とC言語を使用

サンプルプログラム（左：Fortran言語、右：C言語）

```
 program main
      implicit none
C include
      include 'AMTK_f.h'
C fixed value
      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
C interface variable
      integer(4) i,j         ! loop variable
      integer(4) ret         ! return status
      character(len=512) buf ! text buffer
      integer(4) hnd         ! file handle
C meta data
      character(len=512) geo  ! GeophysicalName
      integer(4) num          ! NumberOfScans
C array data
      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
      type(AM2_COMMON_LATLON) LL89a(AM2_DEF_SNUM_HI,LMT) ! latlon for 89a
      real(4) tb06h(AM2_DEF_SNUM_LO,LMT)  ! tb for 06h
```

```
#include <stdio.h>
#include <stdlib.h>
#include "AMTK.h"
// fixed value
#define LMT 2200 // limit of NumberOfScans
int main(int argc, char *argv[]){
  // interface variable
  int i,j;       // loop variable
  int ret;       // return status
  char buf[512]; // text buffer
  void *vpnt;    // pointer to void
  hid_t hnd;     // file handle
  // meta data
  char geo[512];  // GeophysicalName
  int num;        // NumberOfScans
  // array data
  AM2_COMMON_SCANTIME st[LMT]; // scantime
  AM2_COMMON_LATLON ll89a[LMT][AM2_DEF_SNUM_HI]; // latlon for 89a
  float tb06h [LMT][AM2_DEF_SNUM_LO]; // tb for 06h
```

```
C open
      hnd=AMTK_openH5('test.h5')
```

```
// open
hnd=AMTK_openH5("test.h5");
```

①

```
C read meta: GeophysicalName
      ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
      write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))

C read meta: NumberOfScans
      ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
      read(buf(1:ret),*)num
      write(*,'(a,i12)')'NumberOfScans: ',num
```

```
// read meta: GeophysicalName
vpnt=geo;
ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
printf("GeophysicalName: %s¥n",geo);
// read meta: NumberOfScans
vpnt=buf;
ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
num=atoi(buf);
printf("NumberOfScans: %d¥n",num);
```

②

```
C read array: scantime
      ret=AMTK_getScanTime(hnd,1,num,st)
      write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
     +'time(scan=1): ',st(1)%year,st(1)%month,st(1)%day
     +,st(1)%hour,st(1)%minute,st(1)%second
```

```
// read array: scantime
vpnt=st;
ret=AMTK_getScanTime(hnd,1,num,(AM2_COMMON_SCANTIME **)&vpnt);
printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
, st[0].year, st[0].month, st[0].day, st[0].hour, st[0].minute, st[0].second);
```

③

```
C read array: latlon for 89a
      ret=AMTK_getLatLon(hnd,ll89a,1,num,AM2_LATLON_89A)
      write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89a(pixel=1,scan=1): '
     +,ll89a(1,1)%lat,ll89a(1,1)%lon
```

```
// read array: latlon for 89a
vpnt=ll89a;
ret=AMTK_getLatLon(hnd,(AM2_COMMON_LATLON
        **)&vpnt,1,num,AM2_LATLON_89A);
printf("latlon89a[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", ll89a[0][0].lat,
        ll89a[0][0].lon);
```

④

```
C read array: tb for 06h
      ret=AMTK_get_SwathFloat(hnd,tb06h,1,num,AM2_TB06H)
      write(*,'(a,f9.2)')'tb06h(pixel=1,scan=1): ',tb06h(1,1)
```

```
// read array: tb for 06h
vpnt=tb06h;
ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_TB06H);
printf("tb06h[scan=0][pixel=0]: %9.2f¥n", tb06h[0][0]);
```

⑤

```
C close
      ret=AMTK_closeH5(hnd)
      end
```

```
// close
ret=AMTK_closeH5(hnd);
}
```

⑥

AMSR2データの読み方①

- HDFファイルのオープン
  - HDFファイルオープン関数を使用して、“HDF access file id”（以下ファイルハンドル値）という値を取得する

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定する
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

Fortran言語

変数を宣言する

```
integer(4) hnd
```

ファイルをオープンする

```
hnd=AMTK_openH5(‘test.h5’)
```

- ファイルハンドル値 hndは以後の処理で必要となる

C言語

変数を宣言する

```
hid_t hnd;
```

- hid_tはHDF5ライブラリで定義されている変数

ファイルをオープンする

```
hnd=AMTK_openH5(“test.h5”) ;
```

AMSR2データの読み方②

- ヘッダ部の読み込み
  - メタデータ取得関数を使用
  - ファイルハンドル値、メタデータの名称が必要

sta=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定する
met: メタデータ名称を指定する
out: メタデータ内容が返されます
sta: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

Fortran言語

変数を宣言する

```
integer(4) ret
character(len=512) geo
```

メタデータ（地球物理量名）を読み込む

```
ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
```

C言語

```
int ret;
void *vpnt;
char geo[512];
vpnt=geo;
ret=AMTK_getMetaDataName(hnd,
"GeophysicalName",(char **)&vpnt);
```

- 出力変数はすべてポインタへのポインタとして渡すため、まずポインタを作ってから、そのポインタを渡す
- コンパイラからの警告を避けるために、voidポインタを適切な型にキャスト

Fortran言語

変数を宣言する

```
character(len=512) buf
integer(4) num
```

メタデータ（プロダクト中の観測データのスキャン数）を読み込む
文字として取得されたメタデータを数値に変換しておく

```
ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
read(buf(1:ret),*)num
```

C言語

```
char buf[512];
void *vpnt;
int num;
vpnt=buf;
ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
num=atoi(buf);
```

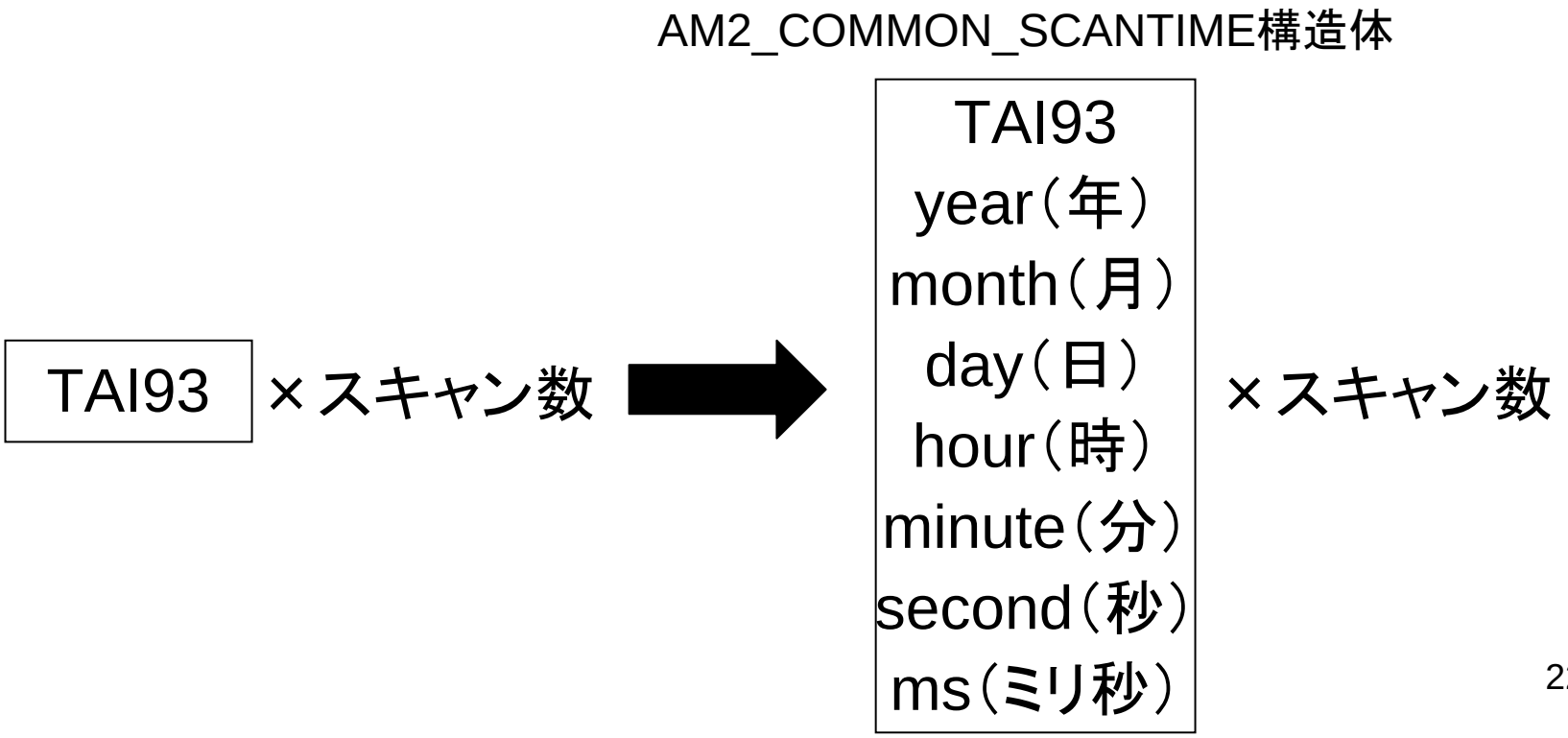
AMSR2データの読み方③

- データ部の読み込み
  - 時刻データには時刻データ取得関数を使用する
  - 時刻データの出力にはAMTKで定義された AM2_COMMON_SCANTIME構造体を使用

sta=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定する
bgn: 開始スキャンを指定する
end: 終了スキャンを指定する
out: 出力データが返されます
sta: [戻り値]失敗の場合は負の値

AMSR2データの読み方③

- AM2_COMMON_SCANTIME構造体
  - AMTK_getScanTimeによって読み込まれた時刻データ（TAI93）はAM2_COMMON_SCANTIME構造体に以下のように格納される



Fortran言語

変数を宣言する

```
integer(4),parameter::LMT=2200
type(AM2_COMMON_SCANTIME) st(LMT)
```

時刻データ（1スキャン目からnumスキャン目まで）を読み込む

```
ret=AMTK_getScanTime(hnd,1,num,st)
```

C言語

- ノミナルスキャン数は約1979
- スキャン数の上限LMTを2200に設定

```
#define LMT 2200;
AM2_COMMON_SCANTIME st[LMT];
vpnt=st;
ret=AMTK_getScanTime(hnd,
1,num,(AM2_COMMON_SCANTIME **)&vpnt);
```

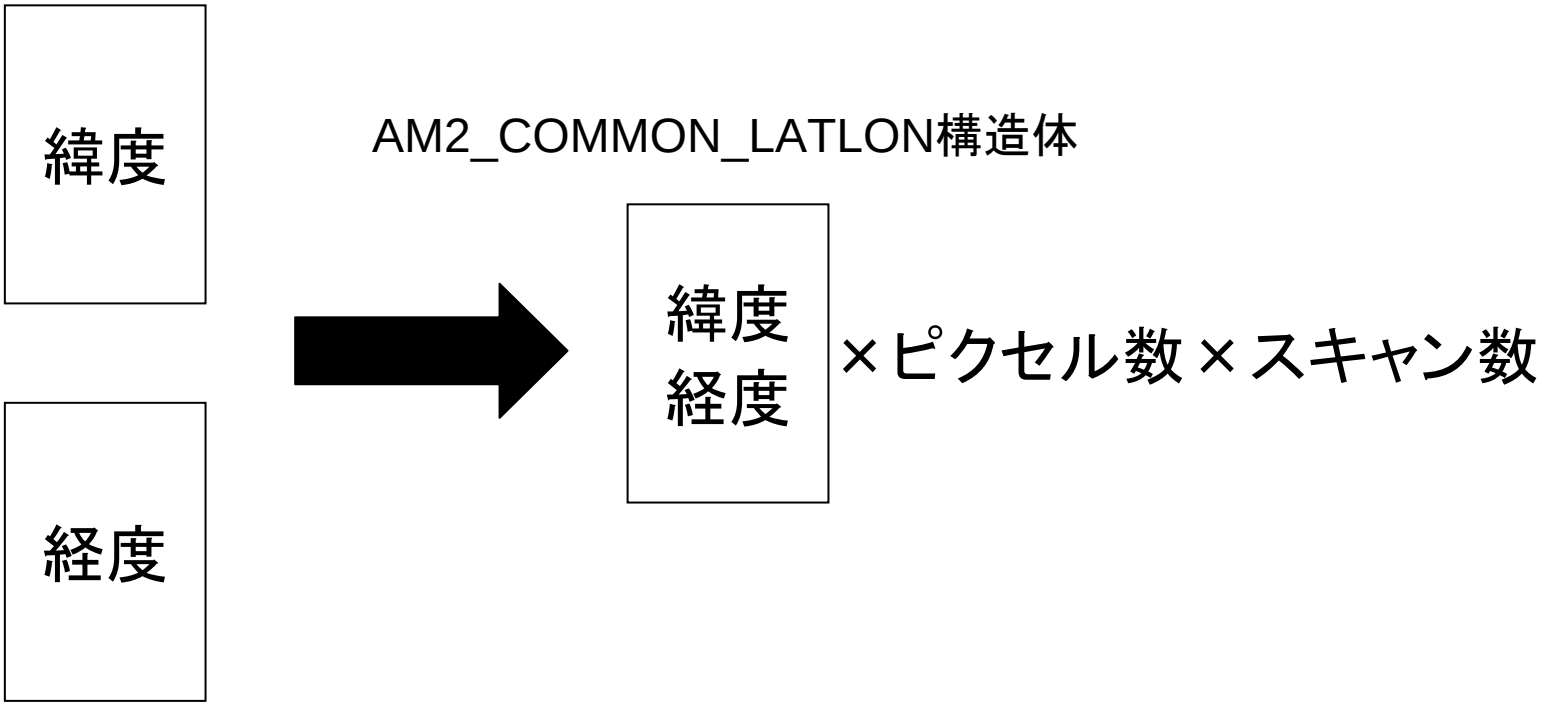
AMSR2データの読み方④

- データ部の読み込み
  - 緯度経度データには緯度経度データ取得関数を使用する
  - 緯度経度データの出力にはAMTKで定義されたAM2_COMMON_LATLON構造体を使用

sta=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定する
out: 出力データが返されます
bgn: 開始スキャンを指定する
end: 終了スキャンを指定する
label: **アクセスラベル**を指定する。アクセスラベルはデータ種類によって異なる
sta: [戻り値]失敗の場合は負の値

AMSR2データの読み方④

- AM2_COMMON_LATLON構造体
  - AMTK_getLatLonによって読み込まれた緯度経度データはAM2_COMMON_LATLON構造体に以下のように格納される



Fortran言語

- AM2_DEF_SNUM_HIはAMTKで定義されている変数（観測幅のサンプル数486）

変数を宣言する

```
type(AM2_COMMON_LATLON)
LL89a(AM2_DEF_SNUM_HI,LMT)
```

89A緯度経度データ（1スキャン目からnumスキャン目まで）を読み込む

```
ret=AMTK_getLatLon(hnd,LL89a,1,num,AM2_LATLON_89A)
```

C言語

- AM2_LATLON_89Aは89A緯度経度データ用のアクセスラベル

```
AM2_COMMON_LATLON LL89a[LMT][AM2_DEF_SNUM_HI];
vpnt=LL89a;
ret=AMTK_getLatLon(hnd,
(AM2_COMMON_LATLON **)&vpnt,1,num,AM2_LATLON_89A);
```

AMSR2データの読み方⑤

- データ部の読み込み
  - AMTK_get_Swath～：L1、L2用データ取得関数
  - 読み込むデータによって使用する関数が異なる

sta=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
sta=AMTK_get_SwathInt(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定する
out: 出力データが返されます（Float：4バイト実数、Int：4バイト整数）
bgn: 開始スキャンを指定する
end: 終了スキャンを指定する
label: アクセスラベルを指定する。アクセスラベルはデータ種類によって異なる
sta: [戻り値]失敗の場合は負の値

Fortran言語

- AM2_DEF_SNUM_LOはAMTKで定義されている変数（観測幅のサンプル数243）

変数を宣言する

```
real(4) tb06h(AM2_DEF_SNUM_LO,LMT)
```

6GHz 水平偏波データ（1スキャン目からnumスキャン目まで）を読み込む

```
ret=AMTK_get_SwathFloat(hnd,tb06h,1,num,AM2_TB06H)
```

C言語

- AM2_TB06Hは6GHz 水平偏波データ用のアクセスラベル

```
float tb06h [LMT][AM2_DEF_SNUM_LO];
vpnt=tb06h;
ret=AMTK_get_SwathFloat(hnd,
(float **)&vpnt,1,num,AM2_TB06H);
```

AMSR2データの読み方⑤

- HDFファイルのクローズ
  - HDFファイルクローズ関数を使用して、HDFプロダクトファイルをクローズする

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定する
ret: [戻り値] 失敗の場合は負の値

Fortran言語

ファイルをクローズする

```
ret=AMTK_closeH5(hnd)
```

C言語

ファイルをクローズする

```
ret=AMTK_closeH5(hnd) ;
```

コンパイルスクリプト例（Fortran言語）

- 4～6行目に、インストールしたライブラリの場所を指定する
- 指定したライブラリディレクトリ直下にはincludeディレクトリとlibディレクトリが必要
- 9行目には使用するコンパイラを指定する
- インテルコンパイラ(ifort)またはPGコンパイラ(pgf90)を指定する

```
 1 #!/bin/sh
 2
 3 # library directory
 4 AMTK=/export/emc3/util/Linux-x86_64/AMTK_AMSR2_1.10
 5 HDF5=/export/emc3/util/Linux-x86_64/hdf5_1.8.4-patch1
 6 SZIP=/export/emc3/util/Linux-x86_64/szip_2.1
 7
 8 # fortran compiler
 9 fc=ifort
10
11 # source filename
12 src=test.f
13
14 # output filename
15 out=test_f
16
17 # library order
18 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
19
20 # compile
21 cmd="$fc $src -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include -L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
22 echo $cmd
23 $cmd
```

コンパイル・サンプルプログラム実行例

```
./compile.sh
./test_f
GeophysicalName: Brightness Temperature
NumberOfScans:          1979
time(scan=1): 2012/08/06 18:02:45
latlon89a(pixel=1,scan=1): ( -73.3418,  42.4036)
tb06h(pixel=1,scan=1):    180.63
```

まとめ

- AMTKの特徴および使用例を紹介した
- L2、L3プロダクトもAMTKで入出力可能
- AMTKをより詳しく知りたい
  - AMTKユーザマニュアル
  - AMSR2データ利用解説文書

ご清聴ありがとうございました