# ALMA LO System Setup Algorithms

Stephen Scott

10 June, 2009

Revision 1.9

## 1 Terminology

The following acronyms and terms are used in the description of the LO system:

| | |
|---|---|
| FSKY: | Sky frequency, the desired sky frequency in the center of one baseband. |
| DYTO: | Digitally Tuned YIG Oscillator (this is the 2nd LO); range 8-14 GHz. |
| FTS: | Fine tuned synthesizer, part of the 1st and 2nd LO's (FTS1 & FTS2), range varies from 20-42.5MHz in LO2 or 20-45MHz in LO1. |
| LO2: | Second LO, used to downconvert the first IF bandpass into the 2nd IF. Physically this is the output of a DYTO. Range 8-14GHz. There is an LO2 for each baseband. The DYTO is a harmonic of 125 MHz combined with the FTS2 (20-42.5 MHz), added or subtracted depending on the lock sideband selection, giving discontinuous coverage of the LO2. |
| CVR: | Central Variable Reference, used to generate LS, located in lab. |
| LS: | Laser synthesizer, a component of LO1, distributed from central electronics to antennas. It is the sum of the laser synthesizer multiplier times the CVR minus 125 MHZ |
| LODRIVER: | The driver for the LO1, composed of the sum/ difference of LS and FTS1. |
| LO1: | The first LO downconverts the sky frequency to the IF frequency. The first LO is the product of the cold multiplier and the LODRIVER. |
| LOINT: | The intermediate LO was used as an additional downcoversion step at the ATF and is always an upper sideband conversion. |
| IF: | Intermediate frequency, the absolute difference between the sky frequency and the first LO (after additional subtraction of the intermediate LO when it is used). The IF is receiver band dependent with ranges of 4-8/6-10/4-12 GHz. |
| DIGITIZER: | The aliasing digitizer runs at 4 GHz. |
| IF2: | The second IF band contains the signal band that is passed to the digitizer. The aliasing digitizer runs at 4 GHz with the filtered band just below it, so IF2 is from 4.0-BW to 4.0 GHz (usually 2-4 GHz), with the center at 4.0 - BW/2 => 3.0 GHz. |
| BASEBAND: | The baseband has a center frequency that is equal to the difference of the digitizer frequency and the center of IF2 (typical baseband center 1 GHz). |

## 2 Considerations

### Background

By convention, the LO chain algorithms in this document specify center frequencies of passbands, such as $F_{baseband}$.

To place a sky frequency in the center of one of the basebands the signed sum (the signs depending on the sideband) of $F_{lo1}$, $F_{lo2}$, $F_{loint}$, $F_{digitizer}$, and $F_{baseband}$ must be equal to the sky frequency. Only $F_{lo1}$ and $F_{lo2}$ are variable, reducing the problem to finding the combinations of these two frequencies that yield the desired fixed target frequency represented by the sum/difference of $F_{sky}$ and the fixed frequencies. Note

that placing a sky frequency in the center of the baseband may place the target frequency at a notch in the correlator passband (such as between the tunable filters). Proper placement of the sky frequency should take the correlator configuration into account and would be done by higher level software the controls both the LO chain and the correlator configuration.

SSB and DSB Use Cases

The frontend hardware bands have distinct capabilities for sidebands of the first LO: single sideband, sideband separating, or double sideband. These three types of frontends add complexity to the setup of the LO chain which results in two independent use cases. The first use case is the specification of a single sky frequency for each baseband and can be used for all front end types and is designated as the *SSB use case*. When a double sideband receiver is used it is sometimes desirable to specify sky frequencies for both the upper and lower sidebands of a baseband and this is designated the *DSB use case*.

There is an additional constraint placed by the matrix switch in the IF Processor hardware. This switch routes the USB and LSB inputs to the LO2's to be mixed to form the basebands. These switches routes the USB/LSB inputs to ***pairs*** of LO2's. As a result the pairs Baseband0/Baseband1 and Baseband2/Baseband3 must each share the same sideband.

Frequency Errors

When more than two sky frequencies are specified the discontinuous coverage of $F_{lo2}$ may not allow all sky frequencies to be set without error. This applies to both use cases. Additionally, the DSB setup with sky frequencies specified in both sidebands of a single baseband provides sufficient constraints such that the first LO frequency is essentially determined with no flexibility in IF placement. In this case sky frequencies should only be chosen in a single sideband of the remaining basebands or large errors may result. The user can control which sky frequencies are more important in how the errors are apportioned by using a weight that is associated with each sky frequency.

Sideband Preferences

In most cases the software can be allowed to choose the first LO sideband based on the capabilities of the frontend. This would include cases where there are legitimate solutions available for both sidebands. For some other SSB setups it is desirable to specify the sideband of the sky frequency, perhaps to avoid a strong line in the suppressed image sideband, or to avoid a problematic range of the first LO. The software must support these different cases by allowing for a sideband preference to be specified.

IF Frequency

The frontend may have parts of the IF band where performance is significantly better (this is especially true for prototype frontends), requiring that the user have the ability to specify the preferred IF frequency. A default value of the center of the IF band will allow many users to ignore this specification. Because of the constraints present in the DSB use case any IF frequency specifications are ignored.

## Operational Constraints

There will be times when some of the hardware does not work, or does not work correctly, particularly during commissioning. To ensure complete flexibility in these cases specific sky frequencies must be associated with specific baseband hardware, including skipping unused hardware basebands. In general, tuning solutions must be computed for all lock sideband combinations of $F_{fts1}$, $F_{fts2}$, and $F_{lo1}$ (when there is a choice) even when some of these locks have larger frequency errors. These solutions

may be further sorted based on operational constraints, and those that are not supported or discouraged by the current hardware configuration can be removed.

### Algorithm Overview

The basic algorithm is driven by a table reflecting the System Block Diagram which contains all of the relevant hardware parameters for all bands. The experience at the ATF shows that there is a need to allow for customization of both the hardware capabilities and the algorithm until the hardware and software have reached stability. This is done using inheritance, where new behavior simply over-rides the base implementation without changing the base implementation or the interface. Different hardware capabilities are supported by over-riding the method(s) that return parameter values that describe the capabilities. Specific algorithm steps can be over-ridden in the same way.

The LO chain tuning procedure begins with an immutable core algorithm that finds as many tuning solutions as possible for the hardware specifications, including all possible lock sidebands of FTS1 and FTS2. The next step assigns each solution a score, using a simple algorithm based on the sky frequency errors and distance from the preferred IF frequency, which can be over-ridden for a given installation. This is followed by editing, which can also be over-ridden, to remove solutions that are disallowed by some restrictions on the current hardware configuration. Finally the best score is selected from the remaining solutions. All solutions after editing are preserved, along with the score, so that they can be retrieved by a client and advanced selection done there. This procedure allows for maximum flexibility.

The behavior of the algorithm at the edge of each receiver band has been the subject of discussion. The algorithm requires that the full baseband be contained within the frequency span of the receiver. While it is arguable whether the baseband should be allowed to partially extend beyond the receiver band frequency limits, it is in fact the specified range of the LO2 and LO driver for each band that are the fundamental restriction. These frequencies are specified in the System Block Diagram such that the baseband must be fully contained within the receiver frequency span.

## 3 Algorithm Details

The algorithm differs significantly for the SSB and DSB use cases. The SSB algorithm will produce many solutions, while a DSB setup will produce only a few. In both cases the first step is basic parameter checking.

### Parameter Checking

The first step in the algorithm is checking the input parameters for basic plausibility. If inconsistencies are found then exceptions are thrown. If input values are omitted then defaults will be supplied. Input sky frequencies must fall within a frontend receiver band (actually so that the baseband will be completely inside the front end range), and all specified sky frequencies must correspond to the same band. The Band2/Band3 overlap is resolved if necessary. If sideband preferences are input they are compared against the physical capabilities of the frontend, otherwise they are automatically assigned. Sideband preferences are also checked against the IF processor constraint that the first two basebands must be from the same sideband and the last two basebands must be from the same sideband.

Solution Generation

The discontinuous coverage of $F_{lo2}$ provides constraints in the setup of the LO chain when trying to center desired sky frequencies in the basebands. The second LO is also used for fringe tracking and frequency offsets for sideband separation and suppression that further reduce the range by one MHz at each end. This reduction is referred to as the *offset frequency guard*. In the SSB use case, for any single baseband there are many combinations of the two variable LO's that will map to the desired frequency. In this case, the FTS2 can be placed in the center of its range and $F_{lo1}$ chosen to give the correct sky frequency. Two basebands provide enough constraints that it may not be possible to center both of the sky frequencies in the baseband. The maximum sum of absolute sky frequency errors using both FTS2 lock sidebands and taking into account the LO offset frequency guards is arbitrarily split between the two basebands so that the sum is 10.75 MHz, giving a weighted average of 5.4 MHz. For three basebands the maximum sum of errors is 21.17 MHz (average 7 MHz). For four basebands the sum of errors is 26.38 MHz (average 6.6 MHz).

For the SSB case, the solution set is composed of all of the harmonics of 125 MHz that give $F_{lo2}$ that will be within its range and yield a valid IF frequency. The $F_{lo1}$ frequency is uniquely determined when the LO2 harmonic is chosen. The $F_{lo2}$ span is the IF bandwidth minus the baseband bandwidth (the baseband must be fully contained within the IF) so half a baseband bandwidth (1 GHz) is lost at both ends. There are 8 LO2 harmonics per GHz, giving 48 (8x6) for an 8 GHz IF and 16 (8x2) for a 4 GHz wide IF. These numbers vary depending on the lock sideband and will be one greater for half the LO2 locks. The solution set is the product of the number of harmonics and the lock combinations, of which there are 4 (FTS1xFTS2). There is a x2 multiplier for each additional baseband (FTS2) that is used for a single sideband receiver.For a two sideband receiver, there would seem to be another x2 from the sidebands but to use both the LOdriver range is usually limited and the restriction of pairing basebands by sidebands also is a limitation. The difference in specified baseband sky frequencies will reduce the range of LO1, reducing the number of possible solutions below the maximum. Basebands with a valid frequency but zero weight will also contribute to the constraints on LO1 and reduce the number of solutions. The 8 GHz wide bandwidth is always paired with a single sideband frontend and the 4 GHz with a double sideband frontend. The maximum number of LO chain solutions for some different combinations of basebands and frontends is shown in Table 1.

| Basebands | One SB (8 GHz) | Two SB (4GHz) |
|---|---|---|
| 1 | 192 | 132 |
| 2 | 390 | 260 |
| 3 | 755 | 516 |
| 4 | 1508 | 1028 |

**Table 1**: Maximum number of LO chain solutions

The main function of the core SSB algorithm is to take each of these tuning solutions and find the $F_{lo1}$ and baseband $F_{lo2}$ values that will give the smallest weighted absolute error, using the weightings assigned to each baseband. This error (the sum of absolute weighted errors for each baseband) and a 'score' are included with the tuning solution. The highest score is then reported as the preferred solution. It should be noted that the

errors for each baseband are not inversely proportional to the weight as the baseband errors are not completely independent.

The DSB algorithm finds $F_{lo1}$ that will give the smallest weighted sum of errors for each of the lock sideband combinations. The sky frequency errors for a DSB baseband are a function of the difference in frequency of the upper and lower sideband frequencies and the LO2 coverage and cannot necessarily be reduced to zero by moving $F_{lo1}$. The number of solutions derives from lock sideband combinations and is 2 raised to the power of the number of basebands plus one, with the additional one arising from FTS1. SSB baseband specifications may be mixed with a DSB spec, but the frequencies should be chosen judiciously as a single DSB specification essentially fixes $F_{lo1}$.

### LO1

$F_{lo1}$ is the sum of the $F_{cvr}$ times an integer multiplier, a fixed offset, and $F_{fts1}$. The high resolution of the CVR synthesizer gives $F_{lo1}$ essentially continuous coverage throughout its range, allowing FTS1 to be arbitrarily placed in the center of its range. Independent solutions are chosen for each sideband of the LO1 lock loop (tunehigh=true/false) allowing hardware restrictions to be used later to edit out undesired settings.

### LO2

$\mathrm{F_{lo2}}$, also known as $\mathrm{F_{dyto}}$, is a harmonic of 125 MHz with the $\mathrm{F_{fts2}}$ added/subtracted. The $\mathrm{F_{fts2}}$ has a nominal range of 20 to 42.5 MHz which is reduced by one MHz on each end to allow for antenna based LO offset frequencies, yielding an overall coverage of just over 31%. This discontinuous coverage is the constraint that does not allow all sky frequencies to be set without error.

### Score

The score for each solution in the SSB use case is currently the weighted sum of two terms: the weighted error and the distance of the IF frequency from the requested IF frequency. The requested IF frequency defaults to the center of the IF band when the input IF frequency is zero. In the SSB case the sum of the absolute weighted frequency errors is normalized by 25 MHz. The DSB use case only uses the sky frequency errors, normalized somewhat arbitrarily to 200 MHz, and ignores the IF frequency. The current weighting of the two terms gives a weight of 5 to the weighted frequency error and 1 to the IF distance. Both of these terms are first normalized such that a maximum deviation gives zero and no deviation gives unity. The final scores are scaled so that the maximum is about 10. Scoring is isolated in a single method and should be easy to change or over-ride. The scoring algorithm may require future modifications as more experience is gained with ALMA.

### Hardware variations

The ATF has been a good test case for variabilities in the actual hardware. There is an additional downconversion step (the intermediate LO), and different ranges for the IF and the LOdriver. The **ATFLOsolutions** class inherits from the base **LOsolutions** class and over-rides the methods that return values for the parameters that are different. This allows the base algorithms in LOsolutions to not be polluted by conditional code to handle the case of the ATF. When the ATF goes away, the ATF specific classes can just be deleted. It is anticipated that this pattern may occur again in ALMA and have used the ATF as a way to test the implementation.

The ATF has another constraint that is not easily expressed in parameter values – it wants to use only a single sign of lock for the FTS1. This is handled by over-riding the edit method to remove these solutions. The edit method is called before final scoring is

done, and gives the flexibility to reject solutions for arbitrary reasons. Note that the client can also retrieve all the final solutions, including the score, if desired and apply custom selection criteria to find a solution.

### Band2/Band3 frontend overlap

There is an overlap in the frequency coverage of Band 2 and Band 3 between 84 GHz and 90 GHz. This overlap will not be much of an issue for quite some time as Band 3 is due very early in the schedule and Band 2 much later. Nonetheless, capability must be provided to select the frontend when the frequency is within the overlap. This is done by providing an input parameter that allows a choice of NoPreference as well as an explicit selection of either band. The NoPreference option will trigger an algorithm that currently chooses Band 3 but can be modified in the future as necessary.

## 4 API

### Overview

Structures are defined in IDL that are used as input parameters and to define the LO chain solutions that are returned. These IDL structures are found in ICD/Control. The core of the computational software is in the **LOsolutions** class, a Java module in ICD/SharedCode, which uses the ICD/Control IDL structures for input and output parameters. This code will be used by both the OT and Control, giving a common basis for the LO chain calculations. The solution set is calculated with a single method call containing the requested sky frequencies, weights, sideband preferences and IF frequencies for the basebands. There are two variants of this method covering the SSB and DSB use cases. The frequency overlap between bands 2 and 3 is handled with an additional parameter to support this selection.

There is no automatic mapping of sky frequencies to basebands, allowing full control to the caller of the method. This will allow the case of a broken or missing baseband to be addressed by the caller. For example, use only baseband0 and baseband2. An unused band is specified with a sky frequency less than 1 MHz. This same convention is used to get the default value of the IF frequency of the middle of the band. Valid solutions for all four basebands are always returned even when they are not specified in the input. This is accomplished by copying the solutions for the first valid baseband into the unused or invalid basebands.

The range for the weight values is 0 to 100. While a weight of 0 will remove the baseband from the first LO calculations, the second LO will be correctly calculated as long as there is a valid sky frequency. This is useful for low priority “ride along” basebands.

The result of allowing full flexibility in debugging and commissioning is that there are parameters that will be rarely exercised in a majority of observing. These parameters can be partially hidden from the casual user by default values in Python or OT defaults.

## IDL structures and enums

The following IDL structures are input parameters defined in ICD/CONTROL

```
Enum  SidebandPreference {
    Lower,
    Upper,
    NoPreference
};

struct SSBbasebandSpec {
    double skyFreqHz;
    double weight; // 0 to 100
    double ifFreqHz;
    SidebandPreference sidebandPref;
};

struct BasebandSpec {
    double skyFreqHz;
    double weight;
    double ifFreqHz;
};

struct DSBbasebandSpec {
    BasebandSpec USB;
    BasebandSpec LSB;
};

typedef sequence<DSBbasebandSpec> DSBbasebandSpecSeq;
typedef sequence<SSBbasebandSpec> SSBbasebandSpecSeq;

/**
 *  Only for the limited case of the Band2/Band3 overlap.
 *  Automatic is almost always the correct choice.
 */
Enum  Band2Band3Overlap{
    Band2,
    Band3,
    Automatic
};
```

## LO chain setup computation methods

The following methods are part of the Java LOsolutions class defined in ICD/SharedCode/LOsolutions:

```
/**
 * Single sideband use case, returns number of solutions.
 * @param baseband Vector of single sideband baseband
 * specifications. Specifications are assigned to basebands
 * (hardware) in the order given, first element to baseband0,
 * etc. If a sky frequency is less than 1 MHz then that baseband
 * will not be used in the computations and the solution for
 * that baseband will be a copy of the first valid baseband
 * solution. If the weight is set to zero then the sky
 * frequency will not be used in the first LO calculation,
 * but the 2nd LO will be calculated correctly. The sequence
 * length must be between 1 and 4.
 * @param band2band3pref specify receiver choice when the
 * specified frequency is in the band2/band3 overlap.
 * Almost always the best choice is NoPreference.
 */
short computeSolutions(Vector<SSBbasebandSpec> baseband,
                       Band2Band3Overlap band2band3pref);

/**
 * Double sideband use case, returns number of solutions.
 * The same basic parameters as computeSolutions(), except
 * that a vector of double sideband baseband specifications
 * is used. The parameters have the same use, except that
 * specifying sky frequencies for both upper and lower sideband
 * requires more care. If the same spec is entered for both
 * sidebands of a baseband then the sideband will be
 * automatically chosen.
 */
short computeSolutions(Vector<DSBbasebandSpec> baseband,
                       Band2Band3Pref band2band3pref);
```

## Tuning parameters

The following IDL structures define the tuning parameters in ICD/CONTROL used as method return values:

```
/// The tuning parameters for one second LO.
 struct LO2Parameters {
    double DYTOFrequency; // used to coarse tune the second LO
    double FTSFrequency; // used to set the FTS
    boolean tuneHigh; // True if the FTS is added to the DYTO.
};

struct TuningParameters {
    /// The receiver cartridge.
    ReceiverBandMod::ReceiverBand receiverBand;
    /// The Central Variable Reference frequency
    double CVRFrequency;
    /// LaserSynth to CVR multiplier
    short LSMultiplier;
    /// First LO Offset Generator (FTS1) Frequency
    double FLOOGFrequency;
    /// True if the FLOOG freq. is added to the 1st LO freq.
    boolean tuneHigh;
    /// The 1st LO driver is multiplied up to give the 1st LO
    /// This is a hardware characteristic and does not set hw.
    short coldMultiplier;
    /// IF processor sideband selection for basebands 0 & 1
    boolean band0band1selectUSB;
    /// IF processor sideband selection for basebands 2 & 3
    boolean band2band3selectUSB;
    /// Tuning Parameters for all four second LO's.
     AllLO2Parameters LO2;
    /// Weighted error; the weighted sum of
    /// abs(actual - requested) frequency over all basebands.
    /// This is one measure of the goodness of the tuning.
    double weightedError;
    /// Score is measure of the relative desirability of tuning
    double score;
    // Tuning solution index, starting at zero
    short index;
};

typedef sequence<TuningParameters> AllTuningParameters;
```

## Utility methods

The following utility methods are part of the Java LOsolutions class defined in ICD/SharedCode/LOsolutions:

```
/// Get number of computed solutions
short getNumSolutions();
/// Get the index of the preferred solution
short getPreferredSolutionIndex();
/// Get the tuning parameters for a specific index
TuningParameters getTuningSolution(short index);
/// Get the preferred tuning parameters
TuningParameters getPreferredTuningSolution();
/// Get all the tuning parameters
AllTuningParameters getTuningSolutions();
/// Get hardware param table as a string (for debugging)
String getHardwareParameters()
```

## Control Interface

The control system uses the CONTROL/ObservingModes/Array/LocalOscillator IDL interface with the following methods implemented in Java. All the utility methods previously mentioned are also available.

```
/**
 * Sets up the entire LO chain for the single sideband use case.
 * This interface wraps computeSolutions as well as sets up all
 * the hardware. See computeSolutions for docs.
 */
short setFrequency(SSBbasebandSpec[] baseband,
                   Band2Band3Pref band2band3pref);
/// Setup LO chain for the DSB use case; see setFrequency().
short setFrequencyDSB(DSBbasebandSpec[] baseband,
                      Band2Band3Pref band2band3pref);
```

Python wraps the LocalOscillator interface and for programmatic use all parameters are available. For interactive use the python interface is smart enough to use many defaults and resolve ambiguities for convenience in CCL testing and debugging.

```
setFrequency(basebandSpecs,
             band2band3pref=Control.BandOverlap.Automatic)
```

The input parameter basebandSpecs is a list of baseband specs, each of which is a list of the parameters for a baseband spec with appropriate defaults. A single input frequency is also supported for convenience. 'None' can be used as a place holder. Parameters that aren't specified take on default values. To avoid ambiguity, a DSB spec needs to be a pair of lists, even if they only contain a single element (the frequency). The interpretation of parameters uses these rules:

- Each baseband is a list, but the list is not required for just a frequency or a frequency and a sideband preference.
- If a baseband contains a list, it is a DSB baseband and must contain two list, one for each sideband. The list with the higher frequency will be assigned to the upper sideband.
- A band overlap can be specified anywhere that a baseband can be specifieid.

Illustrative examples of usage and interpretation:

**setFrequency(x)**

```
  SSB use case
    bb0: f=x, w=100.0, IF=0, sb=NoPreference
```

**setFrequency(x,Control.BandOverlap.Band2,y)**

```
  SSB use case, band2band3pref=Band2
    bb0: f=x, w=100.0, IF=0, sb=NoPreference
    bb1: f=y, w=100.0, IF=0, sb=NoPreference
```

**setFrequency([x,Control.SidebandPreference.Upper,a],y)**

```
  SSB use case
    bb0: f=x, w=a,     IF=0, sb=Upper
    bb1: f=y, w=100.0, IF=0, sb=NoPreference
```

**setFrequency([x,a],[y,b,c])**

```
  SSB use case
    bb0: f=x, w=a, IF=0, sb=NoPreference
    bb1: f=y, w=b, IF=c, sb=NoPreference
```

**setFrequency(x,y,z)**

```
  SSB use case
    bb0: f=x, w=100.0, IF=0, sb=NoPreference
    bb1: f=y, w=100.0, IF=0, sb=NoPreference
    bb2: f=z, w=100.0, IF=0, sb=NoPreference
```

**setFrequency([[U],[L]])**

```
  DSB use case
    bb0: f=U, w=100.0, IF=0, sb=Upper
         f=L, w=100.0, IF=0, sb=Lower
```

**setFrequency(None,None,[[U],[L,40]])**

```
  DSB use case
    bb2: f=U, w=100.0, IF=0, sb=Upper
         f=L, w= 40.0, IF=0, sb=Lower
```

# 5 APPENDIX I: Equations

Fundamental definitions, where $\pm$ or $\mp$ are signs of USB/LSB for LO1 or FTS locks:

$F_{sky} = F_{lo1} \pm (F_{if} + F_{loint})$
$F_{lo2} = N_{lo2} \times 0.125 \pm F_{fts2}$
$F_{ls} = F_{cvr} \times N_{ls} - 0.125$
$F_{lodriver} = F_{ls} \pm F_{fts1}$
$F_{lo1} = F_{lodriver} \times N_{cold}$
$F_{if} = F_{lo2} - F_{if2}$
$F_{if2} = F_{digitizer} - F_{baseband}$
$F_{baseband} = F_{bw}/2$

Other handy equations, where $\pm$ or $\mp$ are still the signs of USB/LSB:

$F_{digitizer} = 4.0$
$F_{bw} = 2.0$
$F_{baseband} = F_{bw}/2 = 1.0$
$F_{if2} = F_{digitizer} - F_{baseband} = 4.0 - 1.0 = 3.0$
$F_{if} = F_{lo2} - F_{if2}$
$F_{if} = F_{lo2} - 3.0$
$F_{sky} = F_{lo1} \pm (F_{if} + F_{loint})$
$F_{sky} = F_{lo1} \pm (F_{lo2} - 3.0 + F_{loint})$
$F_{lo1} = F_{sky} \mp (F_{dyto} \pm F_{fts2} - F_{loint} + 3.0)$
$F_{lo2} = \pm(F_{sky} - F_{lo1}) - F_{loint} + 3.0$
$F_{lodriver} = F_{lo1}/N_{cold}$
$F_{cvr} = \left[\frac{F_{lo1}}{N_{cold}} \mp F_{fts1} + 0.125\right]/N_{ls}$
$F_{lo1} = F_{cvr} \times N_{ls} \times N_{cold}$-0.125$\times N_{cold} \pm (F_{fts1} \times N_{cold})$
$F_{lo1} = F_{lodriver} \times N_{cold}$
$F_{if} = \pm(F_{sky} - F_{lo1})$

# 6 APPENDIX Ii – System Parameters

The basic LO algorithm is driven by the system parameters that describe the hardware capabilities. The parameters for ALMA are hardcoded and then potentially modified if a new system inherits and modifies some of the parameters. The parameters can be returned as a string with the method getHardwareParameters(). The table below is based on the System Block Diagram, Rev 'O'.

| **Band** | **BandFreq** | | **SB** | **IFfreq** | | **Mult** | | **LOdriver** | |
|---|---|---|---|---|---|---|---|---|---|
| | **low** | **high** | | **low** | **high** | **warm** | **cold** | **low** | **high** |
| Band1 | 31.3 | 45.0 | USB | 4.0 | 12.0 | 1 | 1 | 27.3 | 33.0 |
| Band2 | 67.0 | 90.0 | LSB | 4.0 | 12.0 | 6 | 1 | 79.0 | 94.0 |
| Band3 | 84.0 | 116.0 | 2SB | 4.0 | 8.0 | 6 | 1 | 92.0 | 108.0 |
| Band4 | 125.0 | 163.0 | 2SB | 4.0 | 8.0 | 3 | 2 | 66.5 | 77.5 |
| Band5 | 163.0 | 211.0 | 2SB | 4.0 | 8.0 | 2 | 6 | 28.5 | 34.5 |
| Band6 | 211.0 | 275.0 | 2SB | 6.0 | 10.0 | 6 | 3 | 73.7 | 88.3 |
| Band7 | 275.0 | 373.0 | 2SB | 4.0 | 8.0 | 6 | 3 | 94.3 | 121.7 |
| Band8 | 385.0 | 500.0 | 2SB | 4.0 | 8.0 | 6 | 5 | 78.6 | 98. 4 |
| Band9 | 602.0 | 720.0 | DSB | 4.0 | 12.0 | 3 | 9 | 67.8 | 79.1 |
| Band10 | 787.0 | 950.0 | DSB | 4.0 | 12.0 | 6 | 9 | 88.8 | 104.2 |