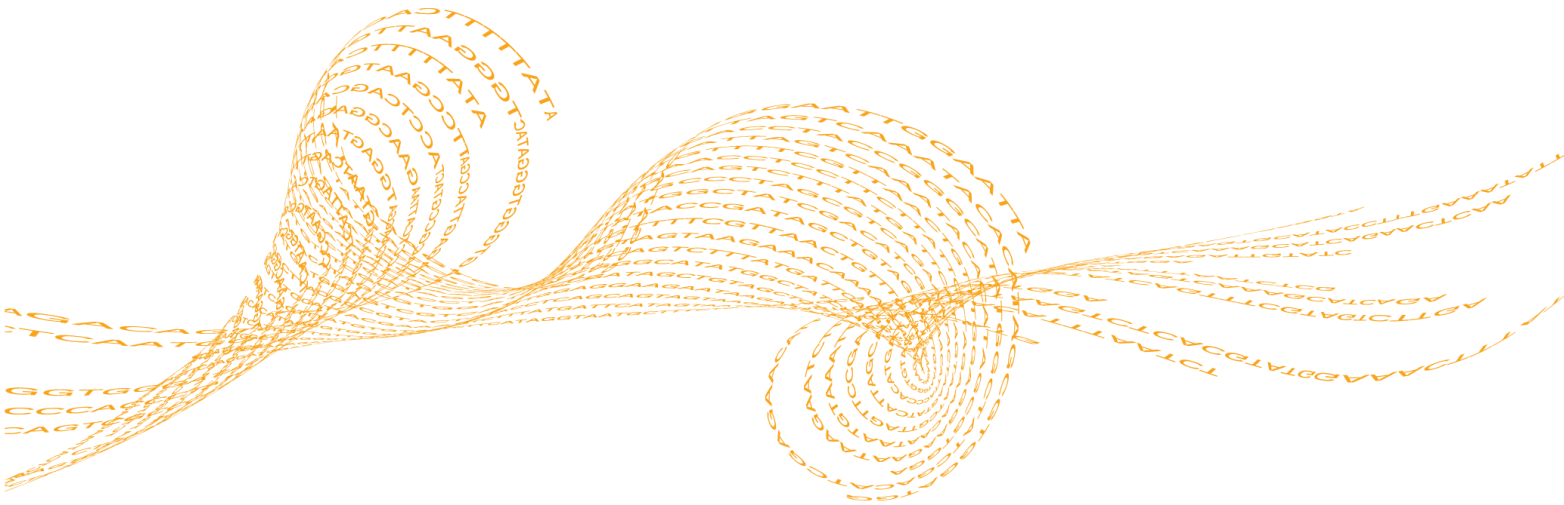
RTA 1.13, HCS 1.5, and SCS 2.10

Theory of Operation

On-Instrument Primary Analysis

FOR RESEARCH USE ONLY



ILLUMINA PROPRIETARY

Pub. No. 770-2009-020, current as of 09 Nov 2011

illumina®

Introduction

RTA is software that helps perform primary analysis for Illumina's sequencing systems. HiSeq® and HiScanSQ™ systems are controlled by the HiSeq Control Software (HCS), while the Genome Analyzer™ system is controlled by the Sequencing Control Software (SCS). HCS performs analysis up to intensity extraction, while RTA performs base-calling and quality scoring. For the Genome Analyzer, SCS handles the first 5 cycles in images analysis, then hands over the image analysis to RTA.

RTA runs locally on the instrument control PC. RTA does not interact directly with the control software, although the control software is configured by default to launch RTA so that the user does not need to interact directly with RTA.

The purpose of this document is to describe the theory of operation for the RTA v1.13 application and the analysis algorithms performed in HCS v1.5 and SCS v2.10.

NOTE
This document does not describe RTA operation on the MiSeq.

In particular, it will answer the following questions:

- How does RTA work, at a high level?
- What are the inputs?
- What are the outputs?
- How does template generation work?
- How does registration and intensity extraction work?
- How are the color matrix and phasing parameters estimated?
- How does base-calling work?
- How does quality scoring work?
- What are the real-time metrics that are produced?
- How are sample preparation controls identified?
- How are errors handled?
- How are data transferred off the instrument PC?

How RTA Works

At the highest level, RTA is simply a state machine. It keeps track of the individual state of each tile (a tile is a section of a lanes that is exactly the size of one image), and when it detects that a tile is ready to advance to the next state, it does the appropriate processing and advances the tile to that state. The conditions to advance the state for a tile are:

- Ready to Pre-Process Cycle1 (performed by HCS, or by SCS during the first 5 cycles)
- Ready to Calculate Template (performed by HCS, or by SCS during the first 5 cycles)
- Ready to Register and Extract Cycle X (performed by HCS, or by SCS during the first 5 cycles)
- Ready to Calculate Color Matrix (normally performed by HCS, or by SCS during the first 5 cycles)
- Ready to Calculate Phasing
- Ready to Base Call Cycle X
- Ready to Quality Score Cycle X

RTA monitors the file system to determine when a tile is ready to advance state. For example, a tile will be ready to Base Call Cycle 15 if:

1 a color matrix has been calculated for that tile
and

2 phasing has been calculated for that tile
and

3 a CIF file exists for that tile for cycle 15, and for all cycles in cycle 15's phasing window

The output of each processing step is always a file, which is then used as a trigger for a subsequent processing step. For example, the output of the extraction step (performed in HCS) is a cluster intensity file (cif), which is then used as a trigger for the base calling step. The output of a base calling step is a base call file (bcl), which is then used as a trigger for the quality scoring step, etc. If a tile is ready to advance in more than one state (say, it's ready to base call cycle 16 and quality score cycle 13), then quality scoring will take priority over base calling.

RTA is multi-threaded and can work with a configurable number of threads (see *Parameters in the Configuration.xml File* on page 21). It is capable of working in the background during a live sequencing run for real-time analysis. The way RTA handles multi-threading is by giving each thread its own subset of tiles for which it is responsible. This minimizes the possibility of thread contention.

When RTA starts up, it will automatically advance each tile's state to the latest possible state, based on which files exist on the file system. In this way, RTA can be shut down and restarted without affecting processing. For example, if RTA starts up and detects that a tile already has its template file (.clocs), and already has its intensity files for the first five cycles (.cif), then it will advance the tile to the "Waiting to Extract Cycle 6" state.

Input

Imaging Data

HCS

A flow cell generates data from the top and bottom surface of each flow cell lane. In addition, up to three swaths can be scanned for each surface, allowing for a maximum of six images per lane. Each image is divided into several "tiles" of manageable size. The number of tiles is configurable in HCS and is 8 by default. Each tile name is a four digit number which encodes the tile's position.

- Highest order digit encodes the surface. 1 is for top, 2 is for bottom
- Second highest order digit encodes the swath. 1 is for first swath, 2 is for second swath, 3 is for third swath
- Remaining two digits encode the tile. 01 through 08 is the default configuration.

For example, the fourth tile from the third swath of the bottom surface has tile number 2304.

SCS

The Genome Analyzer IIx scans 120 tiles per lane by default, divided over two rows of 60 tiles. Each tile corresponds to a unique location where sequencing images (A, C, G, T) were taken. The Genome Analyzer allows only images from one surface, so there is no distinction between top and bottom images.

SCS handles the first 5 cycles in images analysis, then hand over the image analysis to RTA.

RTA Inputs

The primary input files that RTA requires are CIF (cluster intensity file) files (for SCS, the primary input consists of CIFs for the first 5 cycles, and then the image files). The files should reside in the standard HCS or SCS folder structure (e.g., .\Processed\L003\C4.1 for Lane 3, Cycle 4) and with the standard naming convention (e.g., s_5_1101.cif for Lane 5, tile 1101). CIF files contain extracted intensities for all four color channels. The format is described in the appendix.

RTA will also read the RunInfo.xml file which is generated automatically by the control software. This file describes details of the current run. In particular, RTA looks in this file for

- the name of the run
- the number of cycles in the run
- the number of cycles in each read
- whether or not a read is an index read
- the flowcell layout for the run (number of swaths, tiles, etc)

In addition, RTA uses a configuration file Configs\HiSeq.Configuration.xml (or Configs\GA.Configuration.xml) which is detailed in the appendix. Typically, these should not be changed from the defaults.

RTA also uses the SampleSheet to report the indexes count in the InterOp file IndexMetric (see *File Formats and Naming Conventions* on page 25), and has the following characteristics:

- By default RTA reads the file SampleSheet.csv from the run folder. If the file has a different name, then you need to pass command line argument "samplesheet=" when starting RTA (see *Command-line Options for RTA* on page 23).
- RTA can parse both the MiSeq and CASAVA format SampleSheet.
- The file will be copied to the Basecall directory so CASAVA can find it by default.

Finally, RTA can be called with command-line arguments. These arguments are also detailed in the appendix. They include:

- Number of threads to use (defaults to 3)
- Copy Images flag (defaults to false)
- Copy Intensities flag (defaults to false)
- Control Lane (defaults to 0, i.e. none)
- Call Bases (defaults to true)
- ShowGUI (defaults to false)
- Number of cycles
- Instrument name
- Read to analyze (All, 1, 2, 3…)

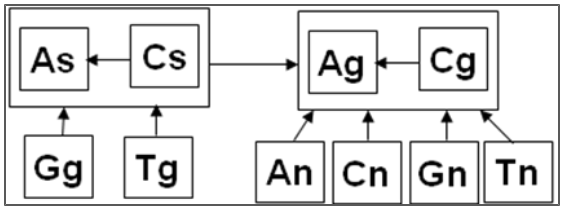
Analysis Steps

Template Generation

The first step in the processing of image data is the generation of templates for each tile. This is performed by HCS or SCS. A template defines the positions of each cluster in the tile, and is used as a reference for the subsequent registration and intensity extraction steps. The templates are defined in a coordinate system relative to the A image of the first cycle.

In the current implementation, template generation requires the first four cycles of image data. Once the fourth cycle for a tile has been imaged, its template can be generated.

Template generation works by finding spots in each image, registering channel A against channel C (counting on the cross-talk between A and C), then registering and merging spots across all cycles to one template. More specifically, the following steps are performed:



1. Find spots in all 16 images (4 channels, 4 cycles), along with the intensity and noise value for each spot
2. Determine the golden cycle g which is the template cycle with the most spots in channels A and C. The silver cycle, s, is the runner up cycle. Image A from the golden cycle (Ag) becomes the frame of reference. Everything else registers against it, directly or indirectly.
3. Register Cg against Ag, and merge Ag and Cg to form reference (A+C)g
4. Merge As and Cs to (A+C)s, and register against (A+C)g
5. Register Gg and Tg against (A+C)s.
6. Register all other images (An, Cn, Gn, Tn) against (A+C)g
7. Merge spot lists together so that each cluster is represented by one (and only one) spot. Spots are ordered by their mean chastity (the minimum ratio of the brightest intensity over the sum of the brightest and second-brightest intensities) on template calls, and HCS or SCS considers spots from best to worst. HCS or SCS also determines whether the lane is high- or low-diversity data by computing the odds that two spots will match calls (agree on all base calls with chastity >0.7) by chance.
8. On high-diversity data, a spot is rejected if (a) it fails the purity filter (chastity <0.6 on two template cycles), (b) the spot is within radius R_1 (1 pixel) of an already-accepted spot, or (c) the spot is within radius R_2 (3.5 pixels) of an already-accepted spot with matching base calls.
9. On low-diversity data, a spot is rejected if (a) it fails the purity filter (chastity <0.6 on two template cyles), or (b) the spot is within the ClusterDistance radius (by default 1.75 pixels) of an already-accepted spot.

10 The template is saved as a locations (.locs) file or a compressed locations (.clocs) file, depending on configuration settings. By default, locations are compressed.

11 Extracted intensities are saved for all template cycles.

It is beneficial to do whatever pre-processing is possible for each template cycle as the images become available so as to minimize the processing time required during template generation. Towards that end, the following steps are performed for each template cycle:

1 Find spots in each of the 4 channels

2 Determine intensity and noise values for all the spots in each image.

3 Save the positions of the spots for each image (.templocs)

4 Save the intensities for the spots (.tempints)

Once the template has been generated, each cycle's images can be registered and extracted against the template. After cycles are registered, offsets are generated. There are small pixel offsets among the four differently colored images taken of each tile, which are due to slightly different optical paths. An .offsets file is created to correct for this, and also corrects for linear rescaling of the image.

Registration and Intensity Extraction

The process of aligning the template of cluster positions onto a given image is referred to as registration, and the process for determining an intensity value for each cluster in the template for a given image is referred to as intensity extraction. For the HiSeq and HiScanSQ, this step is performed by HCS, while for the Genome Analyzer, SCS performs this for the firstfive cycles, while RTA processes the rest of the cycles. For registration, the control software takes advantage of the random nature of the cluster positions by using image correlation to align the template to the image. The basic steps involved with image registration and extraction are:

1 Load the reference template for the current tile from its .locs file or .clocs file

2 Load the TIF files for the 4 channels for the current tile, current cycle

3 For each image:
   a Identify an x, y shift by correlating subregions near the corners of the image to a synthetic image of the template.
   b Use the x,y shift of the four subregions to determine a full 6-parameter affine transformation that transforms the template positions into the image coordinates
   c For each transformed cluster position, use bilinear interpolation to estimate the intensity value of the cluster from a Laplacian pre-sharpened version of the image
   d Subtract background estimated from 32 x 32 pixel regions. Estimates are based on the average of the dimmest four pixels in each region. Subtraction interpolates between regions to remove discontinuities in the estimate.
   e Normalize subtiles of the image such that the 90th percentiles of their extracted intensities are equal. Tiles are divided into a 4 x 4 grid of subtiles
   f Save the array of intensity values for the clusters in a cluster intensity file (cif)

Color Matrix Estimation

A Color Matrix is a 4x4 matrix that is used to correct for the cross-talk between channels. For example, when a cluster lights up in the C channel, some of its light is also collected in the A channel. RTA uses the color matrix to generate matrix corrected intensities which have had this effect reduced or eliminated. The color matrix, M, has entries M_{ij} indicating the amount of observed intensity in channel i generated by signal from nucleotide j.

HCS and SCS perform color matrix estimation during the process of template generation, using cluster intensities from template cycles. After template generation, RTA performs color matrix estimation for each read, using intensities from the first N cycles of the read, where N is the number of template cycles. Color matrix estimation first estimates cross-talk between each pair of channels. The procedure is:

1. Take the (x,y) intensities from these two channels and convert them to polar coordinates (r, θ).
2. Compute a radius-weighted histogram of angles θ in the range [0, 90]
3. Identify the two local maxima, θ_1 and θ_2, in this histogram. (For channels that have no cross-talk, θ_1=0 and θ_2=90). We take $\tan(\theta_1)$ as the matrix element for cross-talk from channel x to channel y, and $\tan(90- \theta_2)$ as the cross-talk coefficient in the opposite direction.

This procedure estimates the matrix entries Mij for i≠j. To complete the calculation of the matrix:

1. Set the diagonal elements of the matrix to 1
2. Correct the intensities using this initial matrix (which orthogonalizes the four color channels), and make preliminary base calls
3. Identify high-chastity (the minimum ratio of the brightest intensity over the sum of the brightest and second-brightest intensities) base calls for each nucleotide.
4. Compute the 10^{th}, 20^{th}, …, 90^{th} percentiles A_1, …, A_9 of the called intensities. Similarly, compute percentiles Ci, Gi, Ti of the called C, G, T intensities.
5. Compute a normalization factor for the C channel, M_{22}, by taking the mean value of A_i / C_i across the percentiles. Similarly for the G and T channels.
6. Finally, normalize the overall matrix to have a determinant equal to 1.

After all tiles have had their color matrix estimated, HCS and SCS compute a median matrix across all tiles and this is the matrix that will be used for the flowcell for the entire run to generate the corrected intensities. The median is calculated element-by-element. However, if a control lane is specified, then HCS and SCS will only use tiles from that lane for calculating the median matrix that will be used for the entire flowcell. The corrected intensities are then calculated by multiplying the observed intensities by the inverse of the color matrix.

Phasing Estimation

RTA assumes that a fixed fraction of molecules in each cluster becomes "phased" at each cycle, in the sense that those molecules fall one base behind in sequencing. If we assume that p is that fraction, then after cycle 1, a given cluster has fraction $(1-p)$ of its molecules on cycle 2, with fraction p still on cycle 1. After cycle 2, $(1-p)^2$ will be on

cycle 3, p(1-*p*) will be on cycle 2, and p^2 will be on cycle 1. In general, after cycle n, the fraction of molecules that will be phased by *k* cycles will be:

$$\binom{n}{k}(1-p)^{n-k}p^k$$

If *p* and *n* are "small" then the intensity contribution from molecules phased more than 1 cycle (second order terms and higher) is small. In that case, we can take the ratio of intensities of molecules that are phased by one cycle and those that are not phased to get:

$$\frac{np(1-p)^{(n-1)}}{(1-p)^n} = \frac{np}{(1-p)} \cong np$$

To estimate *p* in practice, RTA applies the following method for cycles 3 through 12.

1. Correct intensities for cross-talk using color matrix. Filter out clusters with chastity < 0.6 (the minimum ratio of the brightest intensity over the sum of the brightest and second-brightest intensities).
2. For channel "A", determine all clusters in cycle *N* for which "A" is not the brightest intensity. Divide clusters into two groups: clusters where the previous base call in cycle *N*-1 was "A" and clusters where the previous base call was not "A".
3. For clusters where the previous call was "A", it is assumed that each cluster intensity at cycle *N* consists of a phasing component *p* and a noise component *s*, making the total signal *p* + *s*. Normalize each intensity by the intensity of the brightest channel at cycle *N* to get (p + s) / *I* and then average for all clusters in the group.
4. For clusters where the previous call was not "A", it is assumed that each cluster intensity at cycle *N* consists of just a noise component s. Normalize each intensity to get *s* / *I* and then average for all clusters in the group.
5. Set phasing at cycle *N* for channel "A" to the difference between average (p + s) / *I* and average *s* / I.
6. Repeat for all color channels and take average to determine overall phasing at cycle N.
7. Repeat for all cycles and determine best fit line. The slope is the estimated phasing parameter for the entire run.

RTA estimates the pre-phasing parameter q the same way but uses cycles 2 through 11 and compares intensities against cycle *N*+1 rather than cycle *N*-1. If a control lane is specified, only tiles from that lane are used to calculate these parameters. If a read has less than 12 cycles, the phasing and pre-phasing parameters are defaulted to 0.

Once the phasing and pre-phasing parameters are calculated, RTA creates a phasing matrix to model phasing effects. This is done by creating an *N* x *N* matrix where *N* is the total number of cycles. Rows represent cycles and columns represent template termination position. Without phasing or pre-phasing, termination position is expected to match the cycle number at any given cycle. In other words, the probability that the termination position at cycle *n* is equal to position *n* is 1, and 0 elsewhere. For 3 cycles, the matrix would look like:

1	0	0
0	1	0
0	0	1

With phasing and pre-phasing, there are now three probabilities to consider. First, the probability that the position at cycle *n* is equal to *n*-1 is *p*, where *p* is the phasing parameter previously calculated. Second, the probability that the position at cycle *n* is equal to *n* + 1 is *q*, where q is the pre-phasing parameter. Third, the probability that the position matches the cycle number, i.e. position at cycle *n* is equal to *n*, is now 1 - *p* - *q*. Thus, the probability that the position at cycle i is *j*, or **P**(*i*,*j*), is the sum of 3 contributing probabilities:

- *p* * **P**(*i*-1, *j*): the probability that phasing occurred; position did not change from previous cycle
- (1-*p*-*q*) * **P**(*i*-1,*j*-1): the probability that no phasing or pre-phasing occurred; position incremented 1 from previous cycle
- q * **P**(*i*-1,*j*-2): the probability that pre-phasing occurred; position incremented 2 from previous cycle

With this definition, RTA builds the following phasing matrix.

	Pos *j* = 1	Pos *j* = 2	Pos *j* = 3	...	Pos *j* = *N*
1	(1-*p*-*q*)	*q*	0	...	0
2	*p* * **P**(1,1)	(1-*p*-*q*) * **P**(1,1) + *p* * **P**(1,2)	q * **P**(1,1) + (1-*p*-*q*) * **P**(1,2) + *p* * **P**(1,3)	...	0
...	...	...	...	...	...
N	*p* * **P**(*N*-1,1)	(1-*p*-*q*) * **P**(*N*-1,j-1) + *p* * **P**(*N*-1,*j*)	q * **P**(*N*-1,j-2) + (1-*p*-*q*) * **P**(*N*-1,j-1) + *p* * **P**(*N*-1,*j*)	...	q * **P**(*N*-1,*N*-2) + (1-*p*-*q*) * **P**(*N*-1,*N*-1) + *p* * **P**(*N*-1,*N*)

To phase correct intensities for a given cycle, RTA takes the inverse of the phasing matrix and extracts the matrix row corresponding to the cycle. Probabilities less than a threshold of 0.1 are set to 0, thus creating a phasing window which is applied to the vector of observed intensities values. Essentially, the vector of actual intensities for cycles 1 through *N* is the product of phasing matrix inverse and observed intensities for cycles 1 through N.

$I_a = M^{-1} \times I_o$

Base Calling

Base calling refers to the process of determining a base call (A, C, G, T) for every cluster of a given tile at a specific cycle. In order to base-call, intensities must be corrected for channel cross-talk and for phasing and pre-phasing. The pre-phasing correction implies that base calling will always lag intensity extraction, as knowledge of future cycles' intensities are required in order to correct for pre-phasing. In addition, base calling takes lower priority than registration and extraction. That is, base calling will not occur for a tile if it is possible to register and extract some subsequent cycle. The reason for this is to remove images from the local hard drive as quickly as possible, since these image files can fill up the local drive. The images are not needed after extraction and so are deleted once extraction occurs.

Base calling itself is a simple procedure. The relevant intensity files for neighboring cycles (determined by the size of the phasing window) are loaded and color-corrected

using the color matrix. Those values are then used to determine a phasing-corrected intensity vector for the current cycle. Each cluster will receive a call based on the brightest phasing-corrected intensity for that cluster.

Once a preliminary base call has been made, RTA applies a refined color matrix, also known as adaptive color matrix. Adaptive matrix can correct for shifts in the relative intensities of the four color channels over the course of the run, or between portions of a tile image. The adaptive color matrix is determined, for a particular tile and cycle, using the following algorithm:

1. Bin clusters into 512x683 regions (configurable) by pixel position
2. For each bin:
   a. Set aside base calls with chastity < 0.7 (the minimum ratio of the brightest intensity over the sum of the brightest and second-brightest intensities)
   b. If any channel has < 5% basecalls out of total basecalls, skip adaptive matrix correction
   c. Calculate the matrix entries based on the median called intensities for each color channel
   d. Orthogonalize and normalize matrix
   e. Apply matrix to all clusters in block

Final base calls are made on the fully-corrected intensities after the adaptive matrix is applied. The bin size used in the adaptive matrix can be modified (or the function can be disabled) through the UseAdaptiveMatrix configuration parameter (see Appendix 1).

Base calls are saved to base call files (bcl) and corrected intensities are saved to dif (dif) files. The dif files are used for quality scoring. The bcl files are binary files with 1 byte per call. The low order 2 bits represent the base call, and the high order 6 bits represent the quality score. Until quality scoring is performed for a tile, the quality score portion of the byte is set to 0. A byte value of zero is reserved for no-calls.

Quality Scoring

Quality scoring refers to the process of assigning a quality score to each base call. The quality score is typically quoted as QXX where the XX is the score and it means that that particular call has a probability of error of 10^(-XX/10). For example Q30 equates to an error rate of 1 in 1000, or 0.1% and Q40 equates to an error rate of 1 in 10,000 or 0.01%.

In RTA, quality scoring is performed by calculating a set of predictors for each base call, and using those predictor values to look up the quality score in a quality table. The quality table is generated using a modification of the Phred algorithm on a calibration data set representative of run and sequence variability. There are six predictors for the model:

- Hexamer Score: Examines hexamers and returns an enrichment factor that reflects how much the hexamer is enriched near the onset of End Anchored Maximal Scoring Segments (EAMSS). EAMSS masks the ends of reads that start with good reliability but transfer to a state of low reliability.
- Motif accumulation: Maintains a cumulative sum of the Hexamer Score predictor, accounting for how difficult the sequence context has been in the prior cycles of the read.
- Penultimate chastity: Measures early read quality in the first 25 bases based on the second worst chastity value.

- Online overlap: Measures the separation between the foreground called intensities and the background intensities.
- Shifted Purity G adjustment: measures the separation of the signal from the noise for the current base call only, while also accounting for G quenching effects.
- Endiness: Tracks how close the read is to completion.

After calculating the quality scores, RTA identifies reads where the second worst chastity in the first 25 base calls is below a threshold, and marks those reads as poor quality data. This is called read filtering so clusters that meet this cutoff are referred to as having "passed filter".

Once the quality score has been determined, RTA will re-save the bcl file, this time with the quality scores encoded in the higher order 6 bits of each byte. RTA will also save a qms file in the cycle directory, which is an empty file that indicates that quality scoring has been performed for that cycle, and it will save a ctr file in the lane directory. The ctr files store values that will be used for subsequent cycle quality scoring, rather than having to recompute those values when the next cycle is quality scored.

Alignment

RTA compares each cluster to a prespecified list of control sequences at cycle 52 of each read (or the last cycle for experiments in which a read has fewer than 52 sequencing cycles), and reports which reads aligned to the control sequences in the control file. The controls are specified in a fasta file distributed with RTA, and are designed to identify specific sample preparation failure modes. The fasta file specifies the control sequences, their identifiers, and the error tolerance required for a match. In particular, the sample preparation controls evaluate A-tailing (CTA), end repair (CTE1 and CTE2), and ligation (CTL). RTA requires that a read have at most 1 difference to a control to be counted as a match, where the difference can be a substitution, insertion, or deletion in the read with respect to the control sequence.

The control aligner is implemented with the Seqan aligner library (http://www.seqan.de). It builds an index structure on the control sequences upon initialization. When reads are presented for inspection, they are first subjected to a fast pre-screening procedure, using the pre-built index in order to determine if the read should be checked in detail. If a read is flagged for further inspection, then it is checked with an efficient verification algorithm.

The pre-build index implements a PEX strategy. The intuition is that if one sequence has at most k differences with respect to another (where k is much less than the sequence length), then there will be exact agreement among short subsequences of the two sequences. For example, consider the following two sequences that differ at two positions.

```
Sequence 1: ATAGGACCAGGATTATA
Sequence 2: ATAGTACCAGGCTTATA
```

Sequence 1 has two substitutions with respect to sequence 2, and three shared subsequences with Sequence 2 (ATAG, ACCAGG, TTATA). This intuition is implemented in an index by recording the locations and identities of short, non-overlapping k-mers at the beginning of each of the control sequences. When a read is pre-screened, the first few non-overlapping k-mers in the read are checked against the index. If enough of them occur in approximately the expected location in a control sequence, then the read is compared to the control using a fast verification. For example, consider following two example sequences, and the corresponding PEX index:

```
Sequence 1: ATACGACGGCCAACCC
```

```
Sequence 2: ACGGTCCGTTCCAGGTTG
```

PEX index: (ATACG: Sequence 1 Position 0), (ACGGC: Sequence 1, Position 5), (ACGGT: Sequence 2, Position 0), (CCGGTT: Sequence 2, Position 5)

If a sequencing read agrees with one of the indexed control sequences (Sequence 1, for instance) to within at most one edit distance error, then it must have either an exact occurrence of ATACG in position zero or position one, or it must have an exact occurrence of ACGGC at one of positions four, five, or six. For additional specificity, RTA indexes and checks an additional k-mer (so for edit distance one, only one exact occurrence is required, but RTA indexes three k-mers and checks for the presence of two).

When a read contains enough k-mers at the expected positions such that it could agree with a control sequence, then the control sequence is aligned to the read using Myer's bit vector algorithm. Myer's bit vector algorithm is an efficient alignment algorithm that uses bit level arithmetic to compute several dynamic programming matrix entries in parallel, and is thus very fast for verification. If the verification algorithm reveals that the read is within the specified error tolerance to the control, then the identity of the match is recorded in the filter file.

In some contexts, a sequencing read will align to multiple controls. In this case, one of the matching controls is chosen at random, and a bit flag is set in the filter file to indicate an ambiguous alignment.

Output

RTA Outputs

The primary output files that RTA produces are bcl and filter files. Each tile that is analyzed will produce a bcl file for each cycle, which contains the base call and associated quality score for every cluster. Each tile also produces a filter file, which specifies whether or not a cluster passed filters. Additionally, RTA will produce stat files which contain aggregated statistics for each cycle, and locs files which contain the x, y position for every cluster. These output files can be used by downstream processes.

RTA can also output intensity files (as generated by HCS or SCS), in the form of cluster intensity files (cif). A single CIF file is generated for every cycle and every tile. CIF files can be used as input to OLB for off-line base calling (see *Restart Analysis Using OLB* on page 16).

Real Time Metrics

RTA provides real-time metrics of run quality in two forms. The first is the Status.htm page within the Data folder. This page offers several views.

- The Run Info view shows general run information such as run time and settings.
- The Tile Status view displays the current processing state and cycle of each tile.
- The Charts view displays average metrics for each tile visually across the physical flow cell. Average metrics include cluster density, % cluster passing filter, and intensity, focus quality, and % quality score greater than Q30 by color and cycle.
- The Cluster Density view displays a box plot representing the distribution of cluster density by lane. The data in this plot is populated as soon as the first cycle is processed. However, during template generation, the number of clusters in the temporary reference is only an estimate (actually, an under-estimate) of the number of clusters in the final template, which is generated in cycle 4. Therefore you will see the values change from cycle 1 through cycle 4. However, they will not change after cycle 4. The data points used to generate this plot are written to file "NumClusters by lane.txt" in the Data/reports directory.
- The Intensity & Focus Quality view displays two box plots. The first plot represents the distribution of 90th percentile raw intensity values grouped by cycle and color channel. This plot will give you an indication of the intensity decay. The data points used to generate this plot are written to file "Intensity by Color and Cycle.txt". The second plot represents the distribution of focus quality grouped by cycle and color channel. The data points used to generate this plot are written to file "FWHM by Color and Cycle.txt" (FWHM stands for full width at half maximum).

The second form of real-time metrics is the InterOp files. InterOp files are binary files containing tile, cycle, and read level metrics. These files can be viewed using the Sequencing Analysis Viewer or parsed directly. The format of these files is specified in the Appendix.

Data Transfer

Data are transferred to their final destination (the output directory) throughout the run on a background thread, which is set to the lowest priority. When a processing

thread wants to copy an output file (e.g., a cif file) to its final destination, it simply saves a .trans file in the Queued directory with a path to the file. The background copy thread uses this as a signal to copy that file to its final destination. In this way, file copying can lag behind processing (due to a slow network, for example) without affecting processing times (at least until the disk fills up).

Error Handling

RTA uses files to determine the state of each tile. In this way, RTA is as robust as possible to power outages or other crashes. When RTA is restarted, it will detect the state of each tile based on the files that exist. Therefore, there should be no concern that RTA is deleting images after it is finished processing them, for example, because by definition, the image won't be deleted until the result files exist. If RTA is restarted, it will detect the presence of the .cif file and continue processing without looking for the original image that was used to create it.

RTA creates a Log.txt file in the Data\RTALogs directory. Whenever an error occurs, it is logged in a separate ErrorLog.txt file. The analysis performed by HCS or SCS is logged in IALog.txt file within the Logs\IALogs subdirectory of the run directory. Errors are logged in IAErrorLog.txt. All four of these files are transferred to the final output destination at the end of processing.

If an exception occurs that prevents successful processing at a step, sometimes RTA will retry, depending on the nature of the exception. If RTA fails after all retries have been exhausted, then RTA will produce the required output file(s) for that step but populate it with blank values. In this way, processing is guaranteed to finish.

Restart analysis from .cifs

In some cases, it is desirable to restart analysis from the extraction step. This re-analysis uses the original .cif files, but revises estimates of phasing and color matrix, and generates new base calls and quality scores). For instance, on a methylation run where the control lane was accidentally not specified, one can specify the control lane and re-analyze from .cifs to get better results.

If you want to restart analysis from cifs, the standard solution is to use the Off-Line Basecaller (OLB). You can also restart RTA. Both options are described below.

Restart Analysis Using OLB

The Off-Line Basecaller (OLB) can perform base calling for the HiSeq, HiScanSQ, or the Genome Analyzer. The standard workflow is to perform base calling using RTA, after which CASAVA performs alignment using the base calling results. If needed, OLB provides the option to perform primary data analysis off-line.

The basic features of OLB are described in the *Off-Line Basecaller User Guide*.

Restart RTA

To run RTA from .cifs:

1. Update the RTA configuration file (HiSeq.configuration.xml or GA.Configuration.xml):
   a. Optional: By default, the working directory for the reanalysis is {Run Folder}\AltPath. To change the working directory, set AlternateProcessedAndDataDirectory (e.g. to c:\temp); clean out any files currently in that folder.

Pub. No. 770-2009-020, current as of 09 Nov 2011

b Make any other required modifications (e.g. ControlLane)

2 Run the RTA executable, and update the UI:
   a Set the "Flowcell Image Directory" to the Images subfolder of the output folder (e.g. c:\basecalls\FlowCellName\Images)
   b Set the "Directory to Save Output Files" (e.g. c:\BaseCalls\Reanalysis)
   c Check the “Analyze Offline From Intensities” checkbox
   d Click the [Start] button

Appendix

HCS/SCS/RTA File Summary

File	Name	Location	Description	Created	Deleted	Trans-ferred	Req for 2nd level analysis
Base call and quality score file	*.bcl	Data\Intensities\ BaseCalls\Lane\ Cycle	Base call and quality score for each cluster with the quality score encoded in the higher-order 6 bits of each byte	Every cycle at base calling and then resaved at quality scoring	End of processing	Y	Y (BCL converter)
Stats file	*.stats	Data\Intensities\ BaseCalls\Lane\ Cycle	Real time statistics for each cycle	Every cycle at base calling	End of processing	Y	Y (BCL converter)
Positions file	*_pos.txt	Data\Intensities	X and Y positions for each cluster	At template generation	End of processing	Y	Y (BCL converter)
Filter file	*.filter	Data\Intensities\ BaseCalls\Lane	Flag indicating whether or not a cluster has passed filter and if it was a control	Cycle 25	End of processing	Y	Y (BCL converter)
Sample sheet	SampleSheet.csv	Run Folder	Describes samples and indices	By user	End of processing	Y	Y
Thumbnail file	*.jpg	Thumbnail_ Images\ Lane\Cycle	Thumbnail image for each tile and color channel	Every cycle at imaging	End of processing	Y	N
Zprof file	*.jpg.zprof	Thumbnail_ Images\ Lane\Cycle	Z height at each line of an image	Every cycle at imaging	End of processing	Y	N
Color Matrix file	*_matrix.txt	Data\Intensities\ BaseCalls\Matrix	Matrix by tile, aggregated by lane, or aggregated across the flowcell	Cycle 4	End of processing	Y	N
Phasing file	*_phasing.txt, *_phasing.xml	Data\Intensities\ BaseCalls\Phasing	Phasing by tile, aggregated by lane, or aggregated across the flowcell	Cycle 12	End of processing	Y	Y (BCL converter)
Phasing by cycle file	*_cycle.txt	Data\Intensities\ BaseCalls\Phasing	Phasing values for first 12 cycles, used to determine phasing estimate	Cycle 12	End of processing	Y	N
Intensity file	*.cif	Data\Intensities\ Lane\Cycle	Raw intensity for each cluster in all four channels	Every cycle at extraction	After base calling	Optional, N by default	Y (if running OLB)
Error map	*.errorMap	Data\Intensities\ Lane\Cycle	Spatial map of error counts in each 512 x 512 block	Every cycle at basecalling	End of processing	Y	N
FWHM map	*.FWHMMap	Data\Intensities\ Lane\Cycle	Spatial map of average FWHM value in each 512 x 512 block	Every cycle at extraction	End of processing	Y	N
Image size	ImageSize.dat	Data	Width and height dimensions of a tile	Cycle 1	End of processing	Y	Y (if running OLB)

File	Name	Location	Description	Created	Deleted	Trans-ferred	Req for 2nd level analysis
Compressed template locations file	*.clocs	Data\Intensities\Lane	Same as pos.txt except compressed and stored in binary format	At template generation	End of processing	Y	Y
Temporary intensity file	*.int	Processed\Lane\Cycle	Intensities from each spot in each channel in cycle 1-4	Cycle 1-4	After template generation	N	N
Temporary spot locs	*.locs	Processed\Lane\Cycle	Spot locations for each channel in cycle 1-4	Cycle 1-4	After template generation	N	N
Transform file	*.xform	Processed\Lane\Cycle 1	Affine transform parameters	At template generation	End of processing	N	N
Focus stats	F_*.txt	Processed\Focus	Focus statistics from HCS/SCS for reporting in RTA	Every cycle at extraction	End of processing	N	N
Intensity stats	I_*.txt	Processed\Intensity	Intensity statistics from HCS/SCS for reporting in RTA	Every cycle at extraction	End of processing	N	N
Corrected intensity file	*.dif	Processed\Lane\Cycle	Intensity for each cluster after matrix and phasing correction	Every cycle at basecalling	After quality scoring	N	N
Alignment file	*.align	Processed\Lane	Flag indicating if a cluster was aligned to PhiX	At cycle 25	End of processing	N	N
Controls file	*.control	Processed\Lane	Flag indicating if a cluster was identified as a control	At cycle 50	End of processing	N	N
Quality scoring flag file	*.qms	Processed\Lane\Cycle	Empty file indicating quality scoring has occurred for this cycle	Every cycle at quality scoring	End of processing	N	N
Quality metrics file	*.ctr	Processed\Lane	Intermediate file that caches quality metrics use for calculation of quality score	Updated at base calling, every 5th cycle	End of processing	N	N
Buffered byte metrics	*.bbm	Processed\Lane	Intermediate file that caches quality metrics use for calculation of quality score	Updated every cycle at quality scoring	End of processing	N	N
Buffered float metrics	*.bfm	Processed\Lane	Intermediate file that caches quality metrics use for calculation of quality score	Updated every cycle at quality scoring	End of processing	N	N
Tile Status	TileStatus*.bin	Processed\TileStatus	Status of tile with cluster density and focus and intensity values for each color channel	Updated every processing cycle	End of processing	Y	N
Transfer Request	.trans	Queued directory	Path to the file to be transferred by the background copy thread	Throughout	When transfer is complete	N	N
Intensities config	config.xml	Data\Intensities	Configuration file describing cycles	Updated every cycle at	End of processing	Y	Y (if running

Pub. No. 770-2009-020, current as of 09 Nov 2011

File	Name	Location	Description	Created	Deleted	Trans-ferred	Req for 2nd level analysis
			imaged	extraction			OLB)
Basecall config	config.xml	Data\Intensities\ Basecall	Configuration file describing cycles basecalled	Updated every cycle at basecalling	End of processing	Y	Y (BCL converter)
RTA configuration	RTA Configuration. xml	Data\Intensities	RTA configuration settings for run	Start of run	End of processing	Y	N
HCS configuration	HiSeqControl Software. Options.cfg	Config	HCS configuration settings for run	Start of run	End of processing	Y	N
Run info	RunInfo.xml	Run folder	RTA run settings	Start of run	End of processing	Y	N
Run parameters	runParamters. xml	Run folder	HCS run settings	Start of run	End of processing	Y	N
RTA start	RTAStart.bat	Config	Script to launch RTA with command line options	Start of run	End of processing	Y	N
Recipe	<fc>.xml	Recipe	Full recipe of run	Start of run	End of processing	Y	N
Recipe state	<fc>_RunState. xml	Recipe	Temporary file to track where the run is relative to the recipe	Updated throughout	End of processing	Y	N
InterOp files	*.bin	InterOp	Binary reporting files for Sequencing Analysis Viewer	Updated throughout	End of processing	Y	N
HCS/SCS log	*.log	Logs	Log of HCS/SCS run events, one file created per cycle	Updated throughout	End of processing	Y	N
HCS/SCS analysis log	IALog*.txt	Logs\IALogs	Log of HCS/SCS analysis events, one file per processing thread plus one file for general events	Updated throughout	End of processing	Y	N
HCS/SCS diag log	*.log	Diag	HCS/SCS diagnostic logs	Updated throughout	End of processing	Y	N
Start up log	Log.txt	Data\RTALogs	Log of RTA start up events	Start of run	End of processing	Y	N
Error log	*_Error.txt	Data\RTALogs	Log of RTA errors	Updated whenever an error is thrown	End of processing	Y	N
Copy log	*_Copy Thread. txt	Data\RTALogs	Log of RTA copy events	Updated throughout	End of processing	Y	N
Delete log	*_Delete Thread. txt	Data\RTALogs	Log of RTA delete events	Updated throughout	End of processing	Y	N
General log	*_Other.txt	Data\RTALogs	Log of RTA general events	Updated throughout	End of processing	Y	N
Processing log	*_Processor *. txt	Data\RTALogs	Log of RTA processing events	Updated throughout	End of processing	Y	N
Status Summary	Status.xml	Data	Processing status and real time metric charts, uses status.xsl style sheet in same dir	updated throughout	End of processing	Y	N

File	Name	Location	Description	Created	Deleted	Trans-ferred	Req for 2nd level analysis
Status Template files	*	Data\ Status_Files	Template files (htm, xsl, js) for status page	Start of run	End of processing	Y	N
Status data files	*	Data\reports	Data files and images for status page	Updated throughout	End of processing	Y	N
Offsets summary	Offsets.txt	Data\Intensities\ Offsets	Offsets for every tile, every channel, every cycle	Updated every cycle at extraction	End of processing	Y	N
SubTile Offsets	SubTileOffsets. txt	Data\Intensities\ Offsets	Subtile offsets for every tile, every channel, every cycle	Updated every cycle at extraction	End of processing	Y	N
Image analysis complete flag	ImageAnalysis_ Netcopy_ complete*	Run folder	File indicating that processing has completed for a read	End of read processing	End of processing	Y	N
Basecall complete flag	Basecalling_ Netcopy_ complete*	Run folder	File indicating that all post processing has completed for a read	End of read post processing	End of processing	Y	N

Parameters in the Configuration.xml File

Option	Description	Default
NumberOfThreads	Number of processing threads to use.	4
NumberOfLanes	Number of lanes in a flowcell.	8
TilesPerLane	Number of tiles in a lane.	32 for HiSeq 120 for GA
ControlLane	Which lane is to be used as a control lane for Matrix and Phasing estimation.	0 (indicates no control lane)
PostProcessEventFile	Allow a batch file or exe to be called at the end of RTA analysis. Arguments are last extracted cycle number, local run directory, output directory, and read type. Read type distinguishes which read has completed.	No command (disabled)
MinimumDiskSpaceGB	Minimum disk space on the image drive before RTA will pause processing.	1 GB free disk space
CopyIntensities	Copy CIF files to the output directory.	false
ClusterDistance	Nominal distance between clusters.	1.75 (set in HCS)
ClusterFWHM	Cluster full width at half maximum.	0 (set in HCS)
DetectionThreshold	Detection threshold value.	3.5 (set in

Pub. No. 770-2009-020, current as of 09 Nov 2011

Option	Description	Default
		HCS)
FirstTemplateCycle	First cycle to start template generation.	1
TemplateCycleCount	Number of cycles to use for template generation.	4 for HiSeq 5 for GA
QualityScoreType	Quality score algorithm.	v6 for HiSeq v4 for GA
NullRightColumn	Number of pixels to ignore on right of image, these pixels are used for debugging data	6 (set in HCS)
UnmanagedImageAnalysisFlags	Unmanaged image analysis flag.	5
ProcessOnlyLanes	Comma separated list of lanes to process.	None (process all)
ProcessOnlyTiles	Comma separated list of tiles to process.	None (process all)
UseLaneAggregation	Aggregate phasing and matrix data by lane instead of across whole slide. Recommended when no control lane is used.	true
NumAdaptiveMatrixBlocks	Number of sub-tiles to use in adaptive matrix algorithm. Possible values are {0, 1, 4, 9, 12} . Value 0 turns off adaptive matrix.	12
UseFWHM	Use full width half max algorithm.	true (set in HCS)
FixTapBoundaries	Flag to normalize tap boundaries on new images.	true (set in HCS)
AlternateProcessedAndDataDirectory	Alternate location for processed files.	None (disabled)
CopyDIFFiles	Copy DIF files to final network location.	false
ProcessCompleteEventFile	Name of program to run at end of processing completion. Arguments are last extracted cycle number, local run directory, output directory, and read type.	None (disabled)
EnableDiskLocking	Turn on for local disk processing, turn off for network processing of image files.	true
EnableProcessing	Turn on to enable all processing, turn off to stop after image processing.	true
MaximumNumberOfQueuedFiles	Max number of queued files.	10,000,000
AltQualityTable	Alternate Quality Table to use.	None (disabled)

Option	Description	Default
DeleteIntermediateDirectoryAtTheEnd	Delete all intermediate Processed directory after processing.	false
DeleteAllLocalFilesAtTheEnd	Delete all local files after processing.	false
VerifyFilesAfterNetworkCopy	Option to verify contents of files after they are copied to the network.	true
MinutesToDelayImageFileDeletion	Delay deletion of image files to allow manual inspection of images.	0
TransferIntervalInMinutes	Update interval minutes for reporting files.	2
TransferBCLFilesToTheNetwork	Transfer bcl files to network. Also transfers stat files	true
SaveSecondBaseFile	Option to save second base call file	false

Command-line Options for RTA

Command line options are auto-populated by HCS in the RTAStart.bat file. Available options are:

Option	Description	Default
First Argument	Input Directory (expected to be <run folder>\Images)	no default -> must be specified here or in user interface
Second Argument	Output Directory (where the final output will be transferred at the end of processing)	no default -> must be specified here or in user interface
Threads	Number of processing threads to use	2
ShowGUI	Whether or not to show the RTA main window on start up	1 (true)
Read	All, 1, 2, or 3 etc.	All
Control Lane	Control lane	0 (none)
CopyIntensityFilesToNetwork	Whether or not to copy intensity files (cif) to output folder	0 (false)
InstrumentType	Determines which instrument configuration.xml to load	hiseq
AltProcessedDirectory	Directory to use as working directory	none
AnalyzeOfflineFromIntensities	Whether or not to analyze from cifs	0 (false)
CopyImages	Whether or not to copy images	0 (false)
DeleteImages	Delete images after processing	1 (true)
samplesheet	Needed if sample sheet does not have the standard name	SampleSheet.csv

Pub. No. 770-2009-020, current as of 09 Nov 2011

Events

HCS, SCS, and RTA can be configured to run external scripts when an event occurs. This can be done in HiSeqControlSoftware.Options.cfg or *Configuration.xml by setting the event option to the filepath of the script to run.

RTA events are fired with the following arguments:

- Cycle number
- Local run directory
- Output directory
- Read type {"READ1", "READ2", "INDEX"}

For example, for an RTA event you add a line such as this one to the RTA configuration file:

```
<PostProcessEventFile>C:\myScriptFolder\RTA_
   Finished.cmd<PostProcessEventFile/>
```

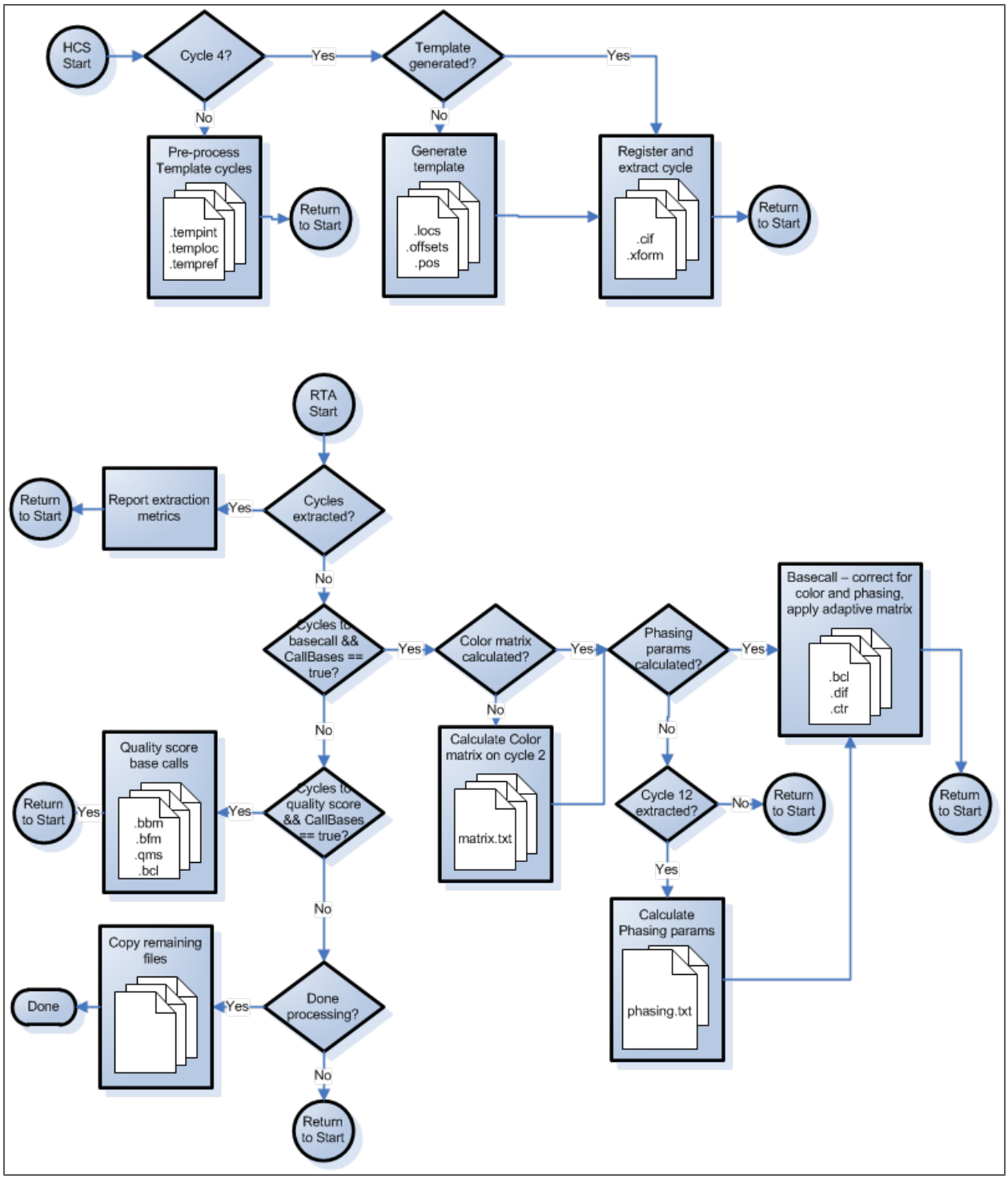
HCS/SCS events are fired with the following arguments:

- Cycle number
- Local run directory
- Output directory
- Run type {"Paired End Indexing Run", "Paired End Run", "Single Read Indexing Run", "Single Read Run", "No Read Run"}
- Event type {"BeginRun", "EndRun", "BeginImagingCycle", "EndImagingCycle", "BeginChemistryCycle", "EndChemistryCycle"}

Event Name	Description	Application
ExtractCycleEventFile	Fired after all tiles have completed an extraction cycle	RTA
BaseCallCycleEventFile	Fired after all tiles have completed a base called cycle	RTA
QualityCycleEventFile	Fired after all tiles have completed a quality scoring cycle	RTA
CIFCycleCopiedCompletedEventFile	Fired after all tiles have copied a cycle's worth of cifs (only fired if cif copy is enabled)	RTA
BCLCycleCopiedCompletedEventFile	Fired after all tiles have copied a cycle's worth of bcls	RTA
PostProcessEventFile	Fired after all tiles have completed processing a read, does not include clean up processing	RTA
ProcessCompleteEventFile	Fired after all processing for the entire run has completed	RTA
StartRunCommandScript	Fired when a run is started	HCS/SCS
EndRunCommandScript	Fired when a run ends	HCS/SCS
StartImagingCommandScript	Fired when the instrument begins imaging a cycle	HCS/SCS
EndImagingCommandScript	Fired when the instrument completes imaging a cycle	HCS/SCS
StartChemistryCommandScript	Fired when the instrument starts chemistry for a cycle	HCS/SCS
EndChemistryCommandScript	Fired when the instrument ends chemistry for a cycle	HCS/SCS

RTA Process Flow

The following diagram shows the general process flow.



File Formats and Naming Conventions

Conventions:

Pub. No. 770-2009-020, current as of 09 Nov 2011

- <lane> is the lane (1..8)
- <read> is the read (1..3 for indexed PE)
- <tile> is the tile (1..120)
- <cycle> is the cycle (1..209 for 101x7x101)

BCL files

The bcl files can be found in the processed directory: <run directory>\Data\Intensities\Basecall\L<lane>\C<cycle>.1 and they are named as follows: s_<lane>_<tile>.bcl. (* note small s)

Format:

bytes 0-3: unsigned 32bits little endian integer: number N of clusters

bytes 4-(N+3): unsigned 8bits integer:

- bits 0-1 are the bases respectively [A, C, G, T] for [0, 1, 2, 3]:
- bits 2-7 contain the quality score (bit 2 is the LSB)
- All bits '0' in a byte is reserved for no-call

Where N is the cluster index

Filter files

The filter files can be found in the directory: <run directory>\Data\Intensities\Basecall\L<lane> and they are named as follows: s_<lane>_<tile>.filter. (*note small letter s).

Format:

Filter files start with the following header.

bytes 0-3: zero value (backwards compatibility)

bytes 4-7: version number (2)

bytes 8-11: number of records

Then each record has 1 byte for the passed-filters flag.

Control files

The control files can be found in the directory: <run directory>\Data\Intensities\Basecall\L<lane> and they are named as follows: s_<lane>_<tile>.control. (*note small letter s).

Format:

Control files start with the following header.

bytes 0-3: zero value (backwards compatibility)

bytes 4-7: version number (2)

bytes 8-11: number of records

Then each record has 2 bytes, representing an unsigned short which indicates the identity of the in-line sample control to which the read was matched, or is zero if the read wasn't identified as a control.

Stats files

The stats files can be found in the directory: <run directory>\Data\Intensities\Basecalls\L<lane>\C<cycle>.1 and they are named as follows: s_<lane>_<tile>.stats.

Byte 0: Cycle = size of integer

Byte 1: Average Cycle Intensity = size of double

Byte 2: All A = size of double
Byte 3: All C = size of double
Byte 4: All G = size of double
Byte 5: All T = size of double
Byte 6: Call A = size of double
Byte 7: Call C = size of double
Byte 8: Call G = size of double
Byte 9: Call T = size of double
Byte 10: Num A = size of integer
Byte 11: Num C = size of integer
Byte 12: Num G = size of integer
Byte 13: Num T = size of integer
Byte 14: Num X = size of integer
Byte 15: Num All A = size of integer
Byte 16: Num All C = size of integer
Byte 17: Num All G = size of integer
Byte 18: Num All T = size of integer

SCL files

The scl files, if configure to be saved, can be found in the processed directory: <run directory>\Processed\L<lane>\C<cycle>.1 and they are named as follows: s_<lane>_<tile>.scl. (* note small s)

Format:

bytes 0-3: 32bit integer: number N of clusters
bits (N * 2 +4) - (N * 2 +5): bases respectively [A, C, G, T] for [0, 1, 2, 3]:
Where N is the cluster index

Pos files

The pos files can be found in the processed directory: <run directory>\Data\Intensities and they are named as follows: s_<lane>_<tile>_pos.txt. (* note small s)

These are text files with 2 columns and <number of cluster> rows. First column is X coordinate and the second column is the Y coordinate. Each line has a <cr><lf> at the end.

Generation of pos files is optional. Template generation will run faster if pos files are disabled.

Locs files

The locs files can be found in the processed directory: <run directory>\Data\Intensities\L<lane> and they are named as follows: s_<lane>_<tile>.locs.

Clocs files

The clocs files are compressed versions of locs file and can be found in the processed directory: <run directory>\Data\Intensities\L<lane> and they are named as follows: s_<lane>_<tile>.clocs.

CIF files

The .cif file format is as follows:

Pub. No. 770-2009-020, current as of 09 Nov 2011

bytes 0-2: CIF
byte 3: Version number (1)
byte 4: Precision. Can be 1 for a file storing intensities as signed bytes, 2 for values stored as signed 2-byte integers, or 4 for values stored as 4-byte floating-point values. Normal .cif files use 2 bytes of precision.
bytes 5-6: Cycle (unsigned short)
bytes 7-8: 1 (unsigned short)
bytes 9-12: Cluster count (unsigned int)
The remainder of the file stores the A intensities, then C, then G, then T. The intensities for each channel take up (Precision * ClusterCount) bytes.

CTR files

The ctr files can be found in the processed directory: <run directory>\Processed\L<lane> and they are named as follows: s_<lane>_<tile>.ctr.

DIF files

The dif files can be found in the processed directory: <run directory>\Processed\L<lane>\C<cycle>.1 and they are named as follows: s_<lane>_<tile>.dif.

QMS files

The qms files can be found in the processed directory: <run directory>\Processed\L<lane>\C<cycle>.1 and they are named as follows: s_<lane>_<tile>.qms.

InterOp files

The interop files can be found in the directory: <run directory>\InterOp.

Extraction Metrics (ExtractionMetricsOut.bin)

Contains extraction metrics such as fwhm (full width at half maximum) scores and raw intensities

Format:

byte 0: file version number (2)
byte 1: length of each record
bytes (N * 38 + 2) - (N *38 + 39): record:
2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: cycle number (uint16)
4 x 4 bytes: fwhm scores (float) for channel [A, C, G, T] respectively
2 x 4 bytes: intensities (uint16) for channel [A, C, G, T] respectively
8 bytes: date/time of CIF creation
Where N is the record index

Quality Metrics (QualityMetricsOut.bin)

Contains quality score distribution

Format:

byte 0: file version number (4)
byte 1: length of each record
bytes (N * 206 + 2) - (N *206 + 207): record:

2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: cycle number (uint16)
4 x 50 bytes: number of clusters assigned score (uint32) Q1 through Q50

Where N is the record index

Error Metrics (ErrorMetricsOut.bin)

Contains cycle error rate as well as counts for perfect reads and read with 1-4 errors

Format:

byte 0: file version number (3)
byte 1: length of each record
bytes (N * 30 + 2) - (N *30 + 11): record:
2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: cycle number (uint16)
4 bytes: error rate (float)
4 bytes: number of perfect reads (uint32)
4 bytes: number of reads with 1 error (uint32)
4 bytes: number of reads with 2 errors (uint32)
4 bytes: number of reads with 3 errors (uint32)
4 bytes: number of reads with 4 errors (uint32)

Where N is the record index

Tile Metrics (TileMetricsOut.bin)

Contains aggregate or read metrics by tile

Format:

byte 0: file version number (2)
byte 1: length of each record
bytes (N * 10 + 2) - (N *10 + 11): record:
2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: metric code (uint16)
4 bytes: metric value (float)

Where N is the record index and possible metric codes are:

code 100: cluster density (k/mm2)
code 101: cluster density passing filters (k/mm2)
code 102: number of clusters
code 103: number of clusters passing filters
code (200 + (N – 1) * 2): phasing for read N
code (201 + (N – 1) * 2): prephasing for read N
code (300 + N – 1): percent aligned for read N
code 400: control lane

Corrected Intensity Metrics (CorrectedIntMetricsOut.bin)

Contains base call metrics

Format:

byte 0: file version number (2)
byte 1: length of each record
bytes (N * 48 + 2) - (N *48 + 49): record:
2 bytes: lane number (uint16)

2 bytes: tile number (uint16)
2 bytes: cycle number (uint16)
2 bytes: average intensity (uint16)
2 bytes: average corrected int for channel A (uint16)
2 bytes: average corrected int for channel C (uint16)
2 bytes: average corrected int for channel G (uint16)
2 bytes: average corrected int for channel T (uint16)
2 bytes: average corrected int for called clusters for base A (uint16)
2 bytes: average corrected int for called clusters for base C (uint16)
2 bytes: average corrected int for called clusters for base G (uint16)
2 bytes: average corrected int for called clusters for base T (uint16)
20 bytes: number of base calls (float) for No Call and channel [A, C, G, T] respectively
4 bytes: signal to noise ratio (float)

Control Metrics (ControlMetricsOut.bin)

Contains pull out information for Illumina in-line sample controls

Format:

byte 0: file version number (1)
bytes (variable length): record:
2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: read number (uint16)
2 bytes: number bytes X for control name(uint16)
X bytes: control name string (string in UTF8Encoding)
2 bytes: number bytes Y for index name(uint16)
Y bytes: index name string (string in UTF8Encoding)
4 bytes: # clusters identified as control (uint32)

Image Metrics (ImageMetricsOut.bin)

Contains min max contrast values for image

Format:

byte 0: file version number (1)
byte 1: length of each record
bytes (N * 12 + 2) - (N *12 + 13): record:
2 bytes: lane number (uint16)
2 bytes: tile number (uint16)
2 bytes: cycle number (uint16)
2 bytes: channel id (uint16) where 0=A, 1=C, 2=G, 3=T
2 bytes: min contrast value for image (uint16)
2 bytes: max contrast value for image (uint16)

Index Metrics (IndexMetrics.bin and IndexMetricOut.bin)

Reports the indexes count. Format:

Byte 0: file version (1)

Bytes(variable length): record:
2 bytes: lane number(unint16)
2 bytes: tile number(unint16)
2 bytes: read number(unint16)
2 bytes: number of bytes Y for index name(unint16)

Y bytes: index name string (string in UTF8Encoding)
4 bytes: # clusters identified as index (uint32)
2 bytes: number of bytes V for sample name(unint16)
V bytes: sample name string (string in UTF8Encoding)
2 bytes: number of bytes W for sample project(unint16)
W bytes: sample project string (string in UTF8Encoding)

Offset files

The offset file can be found in the Data directory: <run directory>\Data\Intensities\offsets.txt. It gives the coefficients of the affine transformation for each image back to the frame of reference. The columns are:

Lane number
Cycle number
Tile number
Channel number (0,1,2,3 for A,C,G,T)
Zero
shift X (or -99999 for failed registration)
shift Y (or -99999 for failed registration)
scale X
scale Y
shear X
shear Y

SubTileOffsets.txt file

The file Data\Intensities\Offsets\SubTileOffsets.txt gives the measured shift for each quadrant of each image relative to the frame of reference. Normally these two shifts are similar (within 5 pixels) across all four quadrants. If one quadrant doesn't match (due to localized imaging issues), it is discarded. If the offsets from the quadrants do not match, the image fails registration (and has a shifts of -99999 reported in the corresponding entry in the offsets.txt file) The columns are:

Lane number
Cycle number
Tile number
Channel number (0,1,2,3 for A,C,G,T)
Flag (normally 0)
Shift X (top left)
Shift Y (top left)
Score (top left)
Shift X (bottom left)
Shift Y (bottom left)
Score (bottom left)
Shift X (top right)
Shift Y (top right)
Score (top right)
Shift X (bottom right)
Shift Y (bottom right)
Score (bottom right)

Directories

- <run directory>\Data
  - ImageAnalysis_netcopy_complete-<read>.txt
  - Basecalling_netcopy_complete-<read>.txt
  - SubTileOffsets.txt
- <run directory>\Data\RTALogs
  - Log.txt – this is a summary of all important events that happened during the run
  - CopyLog.txt – these are all the files generated by RTA that are copied to the output folder and/or deleted from the run folder
  - Errorlog.txt – This is a list of error that may indicate a potential problem. This file may not exist if the run completed without error. This file should be checked at the end of each run or when abnormal behavior is exhibited by the run.
- <run directory>\Data\TileStatus
  - This holds the current state of each tile. Each file has the form TileStatusL<lane>T<tile>.bin
- <run directory>\Data\reports
  - These are all the files needed to support the run-time status page
- <run directory>\Data\Intensities
  - Config.xml
  - RTAConfiguration.xml – the complete configuration of RTA for this run
  - *SampleSheet.* - The sample sheet for the run if found
  - S_*_pos.txt files
- <run directory>\Data\Intensities\BaseCalls
  - Config.xml
- <run directory>\Data\Intensities\BaseCalls\Matrix
  - S_<read>_matrix.txt – aggregated color matrix for flowcell
  - S_<lane>_<read>_<tile>_matrix.txt – color matrix for tile
  - S_<lane>_<read>_matrix.txt – aggregated color matrix for lane
- <run directory>\Data\Intensities\BaseCalls\Phasing
  - S_<read>_matrix.txt – aggregated color matrix for flowcell
  - S_<lane>_<read>_<tile>_matrix.txt – color matrix for tile
  - S_<lane>_<read>_matrix.txt – aggregated color matrix for lane
- <run directory>\Data\Intensities\Offsets
  - Offsets.txt – contain tile offsets for each cycle and channel relative to the template

Technical Assistance

For technical assistance, contact Illumina Customer Support.

Table 1 Illumina General Contact Information

Illumina Website	http://www.illumina.com
Email	techsupport@illumina.com

Table 2 Illumina Customer Support Telephone Numbers

Region	Contact Number	Region	Contact Number
North America	1.800.809.4566	Italy	800.874909
Austria	0800.296575	Netherlands	0800.0223859
Belgium	0800.81102	Norway	800.16836
Denmark	80882346	Spain	900.812168
Finland	0800.918363	Sweden	020790181
France	0800.911850	Switzerland	0800.563118
Germany	0800.180.8994	United Kingdom	0800.917.0041
Ireland	1.800.812949	Other countries	+44.1799.534000

MSDSs

Material safety data sheets (MSDSs) are available on the Illumina website at http://www.illumina.com/msds.

Product Documentation

If you require additional product documentation, you can obtain PDFs from the Illumina website. Go to http://www.illumina.com/support/documentation.ilmn. When you click on a link, you will be asked to log in to iCom. After you log in, you can view or save the PDF. To register for an iCom account, please visit https://icom.illumina.com/Account/Register.

Illumina, Inc.
9885 Towne Centre Drive
San Diego, CA 92121-1975
+1.800.809.ILMN (4566)
+1.858.202.4566 (outside North America) techsupport@illumina.com
www.illumina.com