

Stellar Compass™ Software User's Manual and Software ICD

Version M1.04
September 11, 2000

Intelligent Decisions, Inc.
888 Nancy Lane
Los Altos CA 94024
650-962-1705
FAX 650-969-2465
email: rick@kiessig.com

REVISION HISTORY

8 Oct 1997	M0.7.1	Revised coordinate system diagram.
12 Jan 1998	M1.0	Revised tunable parameters section. Updated software sizes. New coordinate system diagrams.
8 May 1998	M1.02	Updated the code example. Changed floats to doubles in structures. Removed <code>cmd_status</code> from <code>StellarCompassOutput</code> . Corrected order of <code>res_flags</code> & <code>err_code</code> in struct. Change <code>STARMIRROR</code> to be a Makefile flag. Updated <code>err_resid</code> threshold recommendation. Added info about the <code>WILLTRK</code> flag (ICM only). Added info about <code>est_att</code> field (ICM only). Updated <code>CCDOFFS</code> and <code>CHKSUMBUF</code> . Removed obsolete reference to <code>STAR_ERR_CAMID</code> .
19 July 1998	M1.03	Updated stack space requirements. Removed a lingering reference to <code>cam_id</code> . Corrected the sense of the output quaternion.
6 September 2000	M1.04	Added <code>StStopNow</code> and image-processing limit flags.

OVERVIEW

The Stellar Compass™ software (SCS) is invoked through a single subroutine on the main spacecraft processor. A pointer to a single star image is passed with each call. The SCS does not talk to the operating system or to the star camera.

The SCS reads commands from a structure in global memory. Each star image is scanned for stars, pattern matching is done, and an attitude measurement is calculated. The final result is written to a structure in global memory.

STELLAR COMPASS INPUT

StellarCompassMain is the entry point for the SCS. At least 12000 bytes of stack space should be available when the SCS is called. The SCS does make heavy use of floating point, so the upper-level routines should make sure that the task calling the SCS is set up accordingly¹. Typical acquire-mode run times (not including image acquisition time) are 300 ms or less on a Sun 4/110². The maximum run time on the 7-12-97 field test was 330 ms. To the resolution of the clock on the Sun 4/110, the run time in track mode is typically about 60 ms.

Inputs to the software consist of the following:

```
struct stellar_compass_input {
    Coord3d velocity; /* velocity vector (km/s) for stellar */
                    /* aberration correction; 0 if unknown */
    u_short cmd_flags; /* command flags - see below */
#define STAR_RESET 0x0001 /* forget all previous state */
#define STAR_NORMAL 0x0002 /* normal operational mode */
#define STAR_FRCACQ 0x0004 /* force acquire mode */
#define STAR_ABCORR 0x0008 /* perform aberration correction */
    double focal_length; /* focal length to use (meters) */
    Coord2d ccd_center; /* CCD optical center in pixel coords */
    double rate_ratio; /* ratio of next rate to current rate */
    Quaternion align_q; /* alignment cube correction */
#ifdef ICM
    Quaternion est_att; /* estimated attitude */
#endif
} StellarCompassInput;
```

¹i.e. under VxWorks, the VX_FP_TASK bit should be set

²Although the 4/110 has an 11 MHz clock, its throughput is roughly the same as a 7 MHz CPU, due to its older memory architecture.

To compute an attitude measurement, a simple calling routine might do something like the following:

```
#include <math.h>
#include "fgeom.h"
#include "star.h"
#include "starInt.h"

unsigned char starImage[CCDROWS][CCDCOLS];
unsigned char *rowPtrs[CCDROWS];

for (i = 0; i < CCDROWS; i++)          /* set up row pointers */
    rowPtrs[i] = &starImage[i][0];

/*
 * Calculate an attitude using the image pointed to by
 * the row pointers located in rowPtrs[]. Use the
 * default focal length and optical center.
 */
StellarCompassInput.cmd_flags = STAR_NORMAL | STAR_ABCORR;
StellarCompassInput.velocity.x = 20.0; /* km/sec */
StellarCompassInput.velocity.y = 5.0;
StellarCompassInput.velocity.z = 0.0;
StellarCompassMain(rowPtrs);

/*
 * StellarCompassOutput will now contain the
 * results of the attitude measurement
 */
```

`rowPtrs` is an array of pointers into an image from the star camera. Each pointer in the array points to the beginning of a row of image data. The actual image data is contained in `starImage[][]`. Each row contains `CCDCOLS` pixels. The SCS assumes that the image buffer is full. No effort is made to overlap image processing with image acquisition. The image is scanned starting at `rowPtrs[0]`.

The velocity vector is specified with units of kilometers per second, or zero if it is unknown. The coordinate system used is heliocentric with the X axis pointing toward the vernal equinox, the Z axis pointing in the same direction as the Earth's north pole, and the Y axis forming a right handed coordinate system. It is used by the SCS to correct for stellar aberration. Uncorrected, stellar aberration can result in additional attitude measurement errors on the order of 100 μ r for a vehicle in orbit around the Earth. However, the correction software does require extra run time on the order of 5 to 10 ms per image, in either acquire mode or track mode.

Multiple command flags can be set in `cmd_flags` at once. Details on the operation and meaning of each command are described below:

`STAR_RESET` causes the SCS to forget all previous state and completely reinitialize itself. The next image acquired with the `STAR_NORMAL` command will be processed in acquire mode, since any previous attitude history will be forgotten. All internal data

buffers, queues and lists are reinitialized. This is as close to a power-on reset as the software can do. It should be used only if the software begins producing erroneous results or if it begins behaving otherwise erratically. It is not needed during the normal course of operations. If specified with any other command, `STAR_RESET` takes precedence. `STAR_RESET` is assumed the first time the SCS is called.

`STAR_NORMAL` causes the software to process star images in a default fashion. If the vehicle attitude is previously unknown, the software will operate in acquire mode. After five sequential images have been successfully processed in acquire mode, subsequent images are processed in track mode as long as track can be maintained. Using `STAR_NORMAL` is the only way to enter track mode. Track mode cannot be forced, since the software must first develop knowledge of the vehicle's attitude and attitude rate, and therefore the predicted future star positions, before track mode processing can begin.

`STAR_FRCACQ` causes the next image to be processed in acquire mode. If the software had previously been running in track mode, this command forces acquire mode to be used. Under normal conditions, track mode will continue to run until track is lost. This command should be used if an independent verification of vehicle attitude is ever wanted. It is possible, although highly unlikely, that track mode could get "stuck" delivering incorrect attitude measurements. This command can be used to "clear" the software out if higher level code detects incorrect attitude measurements. Using `STAR_FRCACQ` will cause a new attitude measurement to be made, thereby effectively resetting any previous attitude history. An entire image must be present in memory for this command to work correctly. Swath processing is disabled during acquire mode.

One of either `STAR_NORMAL` or `STAR_FRCACQ` must be set for the SCS to produce an attitude measurement.

`STAR_ABCORR` is used to tell the SCS to perform stellar aberration correction. This command must be combined with either `STAR_NORMAL` or `STAR_FRCACQ`. The correction can only be done if a non-zero heliocentric velocity vector is provided. If the velocity component of the input structure is zero, then aberration correction will not be done, even if `STAR_ABCORR` is specified. At Earth's heliocentric velocity of appx. 30 km/sec, stellar aberration can introduce errors up to appx. 100 μ r if not corrected. The correction requires additional run time on the order of 5 to 10 ms on a 20 MHz CPU.

If the `ICM` compile flag is enabled, `STAR_ESTATT` causes the SCS to use the `est_att` field as the estimated attitude for the current image when the software is operating in track mode. The quaternion is relative to the alignment cube. In acquire mode, this flag has no effect.

The `focal_length` field is used to tell the SCS to use a focal length different than the pre-compiled one. Since an accurate focal length is critical to accurate attitude measurements, it can be set here based on calibration data. A weighted average of focal lengths from a series of images under similar environmental conditions normally provides

the most accurate results. If it is set to zero here, the pre-compiled number will be used instead.

The `ccd_center` field is used to tell the SCS to use an optical center different from the pre-compiled one. Since an accurate optical center is important for accurate attitude measurements, it can be set here based on calibration data. As with the focal length, a weighted average should be generated based on a series of images taken under similar environmental conditions. If this field is set to zero, the pre-compiled numbers will be used instead.

The `align_q` field is used to specify the offset quaternion of the alignment cube. The SCS multiplies the CCD-relative quaternion that it calculates by `align_q` before delivering the final result. If `align_q` is zero, it is changed to the identity quaternion.

For the purpose of attitude propagation, track mode normally assumes that sequential images are acquired at uniform time intervals, in order to predict the location of stars in the current image based on previous attitude measurements. In that case, the `rate_ratio` field should be set to 1.0. If images are ever captured at irregular intervals, `rate_ratio` should be used to specify the ratio of the rate of the current image to the rate of the next image. For example, if the current image was captured 10.0 seconds after the previous one, but the next one will be captured 1.0 seconds after the current one, then `rate_ratio` should be set to $(1.0 / 10.0) = 0.1$.

STELLAR COMPASS OUTPUT

Output consists of the following:

```
struct stellar_compass_output{
    u_short res_flags; /* status of results */
#define STAR_OK 0x0001 /* completed OK */
#define STAR_ACQ 0x0002 /* processed in acquire mode */
#define STAR_TRK 0x0004 /* processed in track mode */
#define STAR_ERR 0x0008 /* error; no match obtained */
#define STAR_BROKE 0x0010 /* broke track */
#define STAR_NOROTS 0x0020 /* ran out of Rotation structs */
#ifdef ICM
#define STAR_WILLTRK 0x0040 /* next img will be in trk mode */
#endif
#define STAR_LIMIT 0x0080 /* image processing limit reached */
    u_short err_code; /* error code */
#define STAR_ERR_NOSTARS 2 /* < 4 blobs in acquire mode */
#define STAR_ERR_NOMATCH 3 /* < 2 total matches */
#define STAR_ERR_NOAGREE 4 /* < 2 agreeing matches */
#define STAR_ERR_TRKSTAR 5 /* < 2 blobs in track mode */
#define STAR_ERR_NOSWATH 6 /* no swaths could be generated */
#define STAR_ERR_STOPPED 7 /* aborted with the StStopNow flag */
    Quaternion q; /* alignment-referenced quaternion */
    double err_resid; /* Quest error residual (radians) */
    u_short num_tri; /* number of triangles used in match */
    u_short num_stars; /* number of stars used in the match */
    Rotation *rot; /* information used for attitude calcs */
} StellarCompassOutput;
```

The `res_flags` field is set to indicate the status of the results being reported. If `STAR_OK` is set, then the rest of the output structure is valid. If it is not set, the SCS could not determine attitude from the image. If `STAR_ACQ` is set, then the image was processed in acquire mode. If `STAR_TRK` is set, then the image was processed in track mode. If `STAR_OK` is set, then either `STAR_ACQ` or `STAR_TRK` will also be set. If `STAR_ERR` is set, then there was an error processing the image. If `STAR_BROKE` is set, then a track mode match failed, and the next image will be processed in acquire mode (the calling routine can pass the same image for immediate re-processing if desired). Track mode will break track only when less than two stars are found in their predicted locations in the image. `STAR_NOROTS` is set to indicate that the pre-allocated Rotation structures were used up during an acquire mode match, and that the match was therefore terminated prematurely. In this case, there will always be less than ten matching triangles, and the stiff mesh check will not be satisfied. If the `ICM` compile flag is enabled, then `STAR_WILLTRK` is set to indicate that the next image will be processed in track mode, when the current image was processed in acquire mode. It is not set during normal track mode operation. If an image-processing limit was reached, the `STAR_LIMIT` flag is set. See the "Tunable Parameters" section of this document for details on image processing limits.

If `STAR_ERR` is set, a code indicating the reason for the failure is set in `err_code`. `STAR_ERR_NOSTARS` indicates that there are less than the four blob minimum required for acquire mode. `STAR_ERR_NOMATCH` indicates that less than two total matches were

generated in acquire mode. `STAR_ERR_NOAGREE` indicates that less than two agreeing matches were generated in acquire mode. `STAR_ERR_TRKSTAR` indicates that less than two blobs were found in track mode. `STAR_ERR_NOSWATH` indicates that no swaths were able to be generated in `StellarCompassSwath` (see below). `STAR_ERR_STOPPED` indicates that processing was aborted with the `StStopNow` flag. If the global variable `StStopNow` is set to 1 while the software is processing an image, all processing will be aborted as quickly as possible. No attempt will be made to produce an attitude measurement, even if partial data is available. If the current image is being processed in track mode, setting `StStopNow` will cause the next image to be processed in acquire mode. This flag would normally be set from another thread to indicate that the SCS has exceeded its allowable time slice, and that it should abort as quickly as possible. Note that while `StStopNow` can be set at any time, its effect will be prompt, but it may or may not be immediate. This flag does not act as an interrupt -- its state is periodically polled by the SCS.

The `q` field is a quaternion representing the attitude of the star sensor with respect to its alignment cube. The CCD-relative quaternion calculated by the software is multiplied by a quaternion representing the attitude of the alignment cube relative to the CCD, as specified in `StellarCompassInput.align_q`. The result is normalized (so that $q_1 \geq 0.0$) and reported in the `q` field. The reported quaternion rotates the actual stars into the alignment cube adjusted “blobs” (star images) -- in other words, it rotates the stellar coordinate system into the camera’s coordinate system.

The `err_resid` field contains an estimate of the error that remains after rotating the blobs into the stars. This is the “residu” parameter return by QUEST. Typical values are approximately $4E-8$. In experiments with field test data, small values of `err_resid` do not correlate well with calculated errors. However, large `err_resid` values did correspond to large calculated errors. Values greater than approx. $0.5E-6$ represent a high probability of a failed match, and should therefore be rejected by upper-level filtering or control software. Values between $0.3E-6$ and $0.5E-6$ might be OK, but should be considered highly suspect, especially when only a few triangles matched (see below).

The `num_tri` field contains the number of triangles found in the matching process. This is another quality measurement metric, in addition to `err_resid` and `num_stars`. The SCS is configured so that the matching process terminates when ten triangles formed from “blobs” match ten triangles from the on-board triangle database, and all of those triangles correspond to approximately the same attitude, or when the matched triangles form a “stiff mesh”, which can happen with as few as four stars. QUEST is then used to produce an aggregate quaternion, using the matching star and blob positions. Less than ten triangles in a match generally indicates a noisy or partially obscured image, and probably degraded measurement accuracy. If the number of triangles drops very low (one to four), there is an increased, although still small, probability that the reported quaternion is incorrect. Five sequential “quality ten” images are needed before the SCS will transition from acquire mode to track mode.

The `num_stars` field is used to report the number of stars used in the attitude measurement. The SCS is configured to use a maximum of five stars when computing attitude. In acquire mode, up to the twelve brightest stars (blobs) are extracted from the image, and they are used 3 at a time in brightness order (220 triangles maximum) until a ten triangle match (which requires five stars) is obtained, or until the bright blobs are exhausted. If the list is exhausted and at least one triangle resulted in a match, the final quaternion generation process is started. When in track mode, as few as two stars can be used. In general, the fewer stars used in a match, the higher the probability that an incorrect attitude is being reported, and the less accurate the reported result (this is not always true; certain conditions can cause accuracy to improve).

The `rot` field points to a `Rotation` structure that contains the information used in the final attitude computation. The information includes pointers to structures describing each star image (`Blob`) used. The description of each `Blob` includes its location and brightness. Pointers to structures describing each star used for the match are also included. These structures describe the location of the star on a unit sphere and its relative instrument magnitude. The catalog number of the star can be computed on the ground based on the address of a `StarEntry` minus the address of the beginning of the star table (`StarDataBase`).

After an image has been successfully processed in track mode, the following structure becomes valid:

```
struct trackswath
{
    int start;
    int end;
} StellarCompassSwath[NSTARS_TRACK];
```

This structure indicates the beginning and ending rows that are predicted to be needed from the next image. The list is sorted by beginning row number, and is terminated by a beginning row number that is less than zero, unless all `NSTARS_TRACK` entries are filled. All of the swaths have been coalesced, so that no two swaths overlap. Rows are numbered from 0 to `CCDROWS-1`.

TYPEDEFS AND STRUCTURE DEFINITIONS

The following typedefs and structure definitions are used above:

```
typedef struct coord2d {
    double x;
    double y;
} Coord2d;
```

```

typedef struct coord3d {
    double x;
    double y;
    double z;
} Coord3d;

/*
 * q1, q2 and q3 represent the x, y and z coordinates of
 * the rotation vector, and q4 is the cosine of the half
 * angle; reported in normal form, with q1 always positive
 */
typedef struct quaternion {
    double q1; /* axis */
    double q2;
    double q3;
    double q4; /* angle */
} Quaternion;

/*
 * Star coordinates on a unit celestial sphere
 */
typedef struct starEntry {
    Coord3d star_u; /* star position on unit sphere */
    double star_mag; /* relative instrument magnitude */
    double star_trackcos; /* SCS internal */
} StarEntry;

/*
 * Star images on the CCD
 */
typedef struct blob {
    struct blob *blob_next; /* SCS internal */
    struct blob *blob_last; /* SCS internal */
    double blob_bright; /* total blob brightness */
    Coord3d blob_u; /* location on unit sphere */
    double blob_ccdx; /* X coordinate on CCD */
    double blob_ccdy; /* Y coordinate on CCD */
    int blob_id; /* SCS internal */
    double blob_dist; /* SCS internal */
    int blob_used; /* SCS internal */
    double blob_weight; /* SCS internal */
} Blob;

```

```

/*
 * Stars and Blobs used for the final attitude calculations.
 * Each pair of StarEntry and Blob triplets describes two
 * matching triangles. Individual stars and blobs can be
 * used more than once, although a given Blob is always
 * paired up with the same star, and vice versa.
 */
typedef struct rotation {
    struct rotation *rotation_next; /* SCS internal */
    Quaternion rotation_quaternion; /* SCS internal */
    double rotation_quality; /* SCS internal */
    int rotation_count; /* SCS internal */
    int rotation_unique_blobs; /* SCS internal */
    StarEntry *rotation_star_ad0[POSFOUND]; /* stars used */
    StarEntry *rotation_star_ad1[POSFOUND]; /* stars used */
    StarEntry *rotation_star_ad2[POSFOUND]; /* stars used */
    Blob *rotation_blob_ad0[POSFOUND]; /* blobs used */
    Blob *rotation_blob_ad1[POSFOUND]; /* blobs used */
    Blob *rotation_blob_ad2[POSFOUND]; /* blobs used */
    Coord3d rotation_unit; /* SCS internal */
} Rotation;

typedef unsigned short u_short;

```

COMPILE AND LOAD DEFINITIONS

The following flags should be used to compile the SCS using the GNU compiler (gcc) on a SPARC platform:

```
-O3
-ansi
-pedantic
-Wall
-Wshadow
-Wpointer-arith
-Wmissing-prototypes
-Wcast-qual
-Wcast-align
-Wwrite-strings
-Wstrict-prototypes
-Wnested-externs
-Winline
-Wtraditional
```

In addition, the following conditional compilation flags should be set (with the “-D” flag):

```
LLNOptics      curved focal plane lens
DOUBLE         use double precision math
float=double   use double precision math
__POSIX_SOURCE ANSI standard header files
__USE_FIXED_PROTOTYPES__ ANSI standard header files
TRACKMORE     use 12 stars & 220 triangles max
AUTOSTREAKTHRESHOLD do bright/dead streak filtering
PIXELBITS8    use 8 bits per pixel
DEVELOP       do not include timer & row chksums
QUEST_OPTIMIZATION calculate final attitude w/QUEST
TRACK_MODE    include track mode code
ALTDCLLEVEL   improved local background calculation
mars          use Mars'98 specific config.
ICM           use ICM specific config (** optional **)
STARMIRROR    use mirroring mode (see below)
```

Two other conditional compilation flags should be used for the ground-only version of the code, but should be turned off for flight code:

```
CALIBRATION    automatic focal length calibration
TRACK_SIM      track mode simulator
```

The flight version should be built using all of the object modules in the “starfile” directory except for starFile.o. A Makefile with the above compilation flags is present in the starfile directory.

The ROM image size of the 01-12-98 version of the software on a Sparc using gcc with the optimizer enabled is 24,816 bytes for star.o and 1,912 bytes for the the

supporting geometry routines, plus 145,736 bytes for starDatabase.o, for a total of 172,464 bytes.

The following flags are recommended when using the Greenhills cross compiler on a SPARC, for the RAD6000 target:

```
-ansi
-cpu=rsc
-nosda
:ppc_option=noparamcount
-Xlcomms
-signedchar
-G
```

Note that bugs have been observed with the Greenhills optimizer, and its use is therefore *not* currently recommended.

In addition, the following conditional compilation flags should be set (with the “-D” flag):

```
LLNLOptics      curved focal plane lens
DOUBLE          use double precision math
float=double    use double precision math
_POSIX_SOURCE   ANSI standard header files
__USE_FIXED_PROTOTYPES__ ANSI standard header files
TRACKMORE      use 12 stars & 220 triangles max
AUTOSTREAKTHRESHOLD do bright/dead streak filtering
PIXELBITS8     use 8 bits per pixel
DEVELOP        do not include timer & row chksums
QUEST_OPTIMIZATION calculate final attitude w/QUEST
TRACK_MODE     include track mode code
ALTDCLLEVEL    improved local background calculation
mars           use Mars'98 specific config
ICM           use ICM specific config (** optional **)
STARMIRROR     use mirroring mode (see below)
CPU=RSC        set the CPU type to RAD6000
```

TUNEABLE PARAMETERS

These following tunable parameters are located in starInt.h:

BRIGHTMAX	12	maximum number of stars examined
TRACKMAXTRI	220	maximum number triangles built from BRIGHTMAX stars
MAXROTATIONS	220	maximum number of allowable Rotations ³

These three parameters can only be changed at compile time. Note that the choice of values here (controlled by the TRACKMORE compile flag listed above) directly (but not completely) controls the maximum run time of the software. The larger these numbers are, the harder the software will try to find a match in acquire mode. Larger numbers can be a good idea in the presence of large amounts of noise, for example. However, larger numbers can also result in long run times for the software. These parameters have been set here under the assumption that it is more desirable to accept a mismatch and to then capture another, hopefully less noisy image, than it would be to allow a potentially long run-time (note that the probability of obtaining an erroneous match also goes up with increased run-time).

For the following parameters, the first column is the name of a global variable that is initialized by the software using the #define constant in the second column. The current value of each constant is shown in the third column. The global variables allow these tunable parameters to be modified at run time by patching the indicated memory location.

StTolerance	TOLERANCE	SCALE (0.0002)	max distance error in triangle side lengths (in radians)
StCcdOffs	CCDOFFS	1	number junk pixels to skip per line
StNoiseIncrement	NOISE_INCREMENT	1	amount by which to change filter tolerance during row scanning
StNoiseMinimum	NOISE_MINIMUM	2	the smallest allowable filter tol
StMaximumEnergy	MAXIMUM_ENERGY	2750	the maximum allowable energy per blob
StAdjDiagDiff	ADJDIAGDIFF	4	adjacent-diagonal leak difference
StScanWidth	SCANWIDTH	5	see below
StNoiseRows	NOISEROWS	2	see below
StNoiseCols	NOISECOLS	4	see below
StChksumBuf	CHKSUMBUF	5	see below

³The recent 1600 image test run from 7-12-97 generated a maximum of 37 Rotations.

StIgnoreEndRows		17	number of rows to ignore at the end of the image
StCutoffSet	CUTOFFSET	2.0	threshold for local DC determination algorithm
StAiryBleed	AIRYBLEED	0.17	flicker noise parameter for leakage
StCentered	CENTERED	2	flicker noise parameter
StStreak255Min	STREAK255MIN	250	minimum pixel value to be considered as being part of an image streak

StAutoLineUp	AUTO_LINE_UP	20	increase per-row tolerance if more than this many hits per row
StAutoLineUpMin	AUTO_LINE_UP_MIN	5	minimum number of hits/row to cause an increase in tol
StAutoLineDown	AUTO_LINE_DOWN	1	decrease per-row tolerance if this many hits per row or less
StNumAcquire	NUM_ACQUIRE	5	the number of sequential successful matches required to transition to track mode
StDpePixelBorders	DPE_PIXEL_BORDERS	5	number of border pixels for track mode
StMaxTolBlobLoc	MAXTOLBLOBLOC	3.0	size of search box for track mode
StQuestFibbl	QUEST_FIBBL	0.000001	Quest parameter
StQuestQuibbl	QUEST_QUIBBL	0.001	Quest parameter
StProbSaturated		400.0	blob brightness assumed by Quest to contain saturated pixels
StSaturatedWeight		1.25e-7	weighting factor applied to saturated blobs by Quest
StNormalWeight		0.001	weighting factor applied to non-saturated blobs by Quest
StMinTrackStars		2	Minimum number of stars required to stay in track mode. Legal values are 2 or 3.
StStopNow		0	Set to non-zero value to abort current processing
StImpulseLimit		0	Maximum number of impulse filtrations
St255Limit		0	Maximum number of saturated pixel processing events
StRotationLimit		0	Maximum number of rotations to process
	SWATH_HALF_WIDTH	6	number rows in half a swath for track mode
	STARMIRROR	on	images are mirrored during processing

STARINVERT	off	images not inverted before processing
CCDMXCENTER	256	default lens mechanical center
CCDMYCENTER	256	
FOCALLENGTH	22.5649e-03	the default lens focal length

Several row and column buffers exist to allow the image processing logic to make excursions as it performs centroiding operations and makes energy distribution decisions. Image processing starts at row number `SCANWIDTH + NOISEROWS`, and ends at the `SCANWIDTH` row before the last row of the image. The first and last columns processed are determined by a circle with its center at `StellarCompassInput.ccd_center`⁴. The first column processed must be at least `CCDOFFS + NOISECOLS + SCANWIDTH` from the left edge, and the last column is at least `NOISECOLS + SCANWIDTH` pixels from the right edge.

After star centroids are calculated, only those that are less than $(ROWWIDTH * 0.5) - SCANWIDTH$ pixels away from `StellarCompassInput.ccd_center` are accepted. If a star centroid extends outside of that limit, the edge of the star image may be distorted and the resulting blob must therefore be discarded.

The most CPU intensive part of the software during normal operation is the initial row scan. The first 4 pixels and the last 4 pixels in each row are overwritten with a special pattern (0,0,255,255). This helps terminate row scanning and backwards 255 scanning. It also means that those columns cannot be used for star images. This is only noticeable for the rows in the middle of the image, where the inscribed circle gets close to the edge of the CCD.

Several parameters are available that can be used to limit the CPU time consumed by the SCS. In their default state, set to 0, these limits are disabled. `StImpulseLimit` can be used to limit the number of impulse filtrations that are processed. If an image exceeds the number, image processing is aborted. However, pattern matching continues as it would have otherwise. If a match is possible given the partial image processing results, the answer is returned, and the `STAR_LIMIT` output flag is set. `St255Limit` works in a similar way, except for saturated pixel processing ("overload") events. `StRotationLimit` can be used to limit the number of different rotations ("triangles") that are examined during the match process. The maximum number of rotations is always `TRACKMAXTRI`, regardless of the value used for `StRotationLimit`.

COORDINATE SYSTEMS

The coordinate system used internally by the software is shown below. Note that starfile's "-b" flag has the same effect on coordinate system transformation as

⁴The starting column is also rounded up to be a multiple of 4, to optimize memory accesses.

STARINVERT. However “-b” also causes the image to be scanned from bottom to top, rather than top to bottom. STARINVERT and STARMIRROR do not change the row scan order.

CCD Image in Memory

