# Sparse Optical Flow (SOF) Demo Algorithm

**September 8, 2017**

# 1. Sparse Optical Flow (SOF) Demo algorithm

This sparse optical flow (SOF) algorithm uses Lucas Kanade tracker as basic tracker (flow vector compute) module. This particular demo algorithm shows creating an SOF algorithm using below 5 applets

- Image/YUV Padding
- Image Pyramid Creation
- Harris corner detection.
- Fast 9 corner Detection
- Best Feature to front using Fast9 Score and applying NMS
- Pyramidal Lucas Kanade tracker

The interface details of each of these applets can be found as part of EVE_Applets_UserGuide located in apps/docs drecotory.

- The SOF algorithm accepts one 8-bit gray scale image for every process call and computes location of keypoints detected/tracked in the prevous frame.
- Since the LK tracker algorithm requires two frames to compute the flow vector, the fist frame supplied to SOF algorithm will be just used for creating image pyramid and will not produce any tracking information.
- The second frame onwards, the flow vector will be computed for the detected/tracked key points using image pyramid of current frame and the previus frame. The image pymaid of current frma will be intaernaly maintained and will be used in the next call as a previus frames with out re-computing again.
- The SOF algorithm supports below three methods for detecting key points
  1. External key points from user
  2. Harris corener detection applet
  3. Fast9 Key point detection along with best feature from applet
- This SOF demo supports a algorithm to track key points across multiple frames.
  - Detects the key points at every user defined number of frames.
  - Filters the key points with error estimates heigher thaa the user defined value.
  - Supress the detected/tracked key points with in user defined window size

The algorithm is supplied with a test bench to demonstrate the usage and interface.

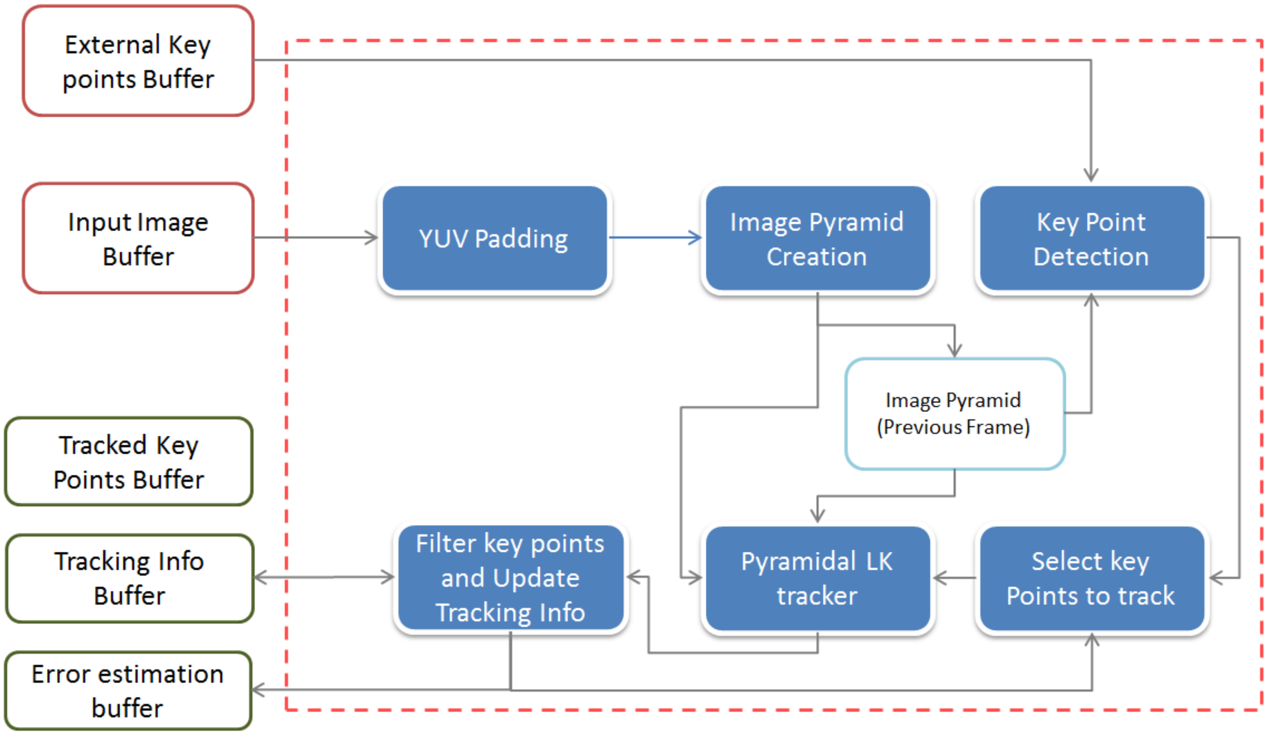
External Key points Buffer
Input Image Buffer
Tracked Key Points Buffer
Tracking Info Buffer
Error estimation buffer
YUV Padding
Image Pyramid Creation
Key Point Detection
Image Pyramid (Previous Frame)
Filter key points and Update Tracking Info
Pyramidal LK tracker
Select key Points to track

**Figure 1 SOF Algorithm Data flow**

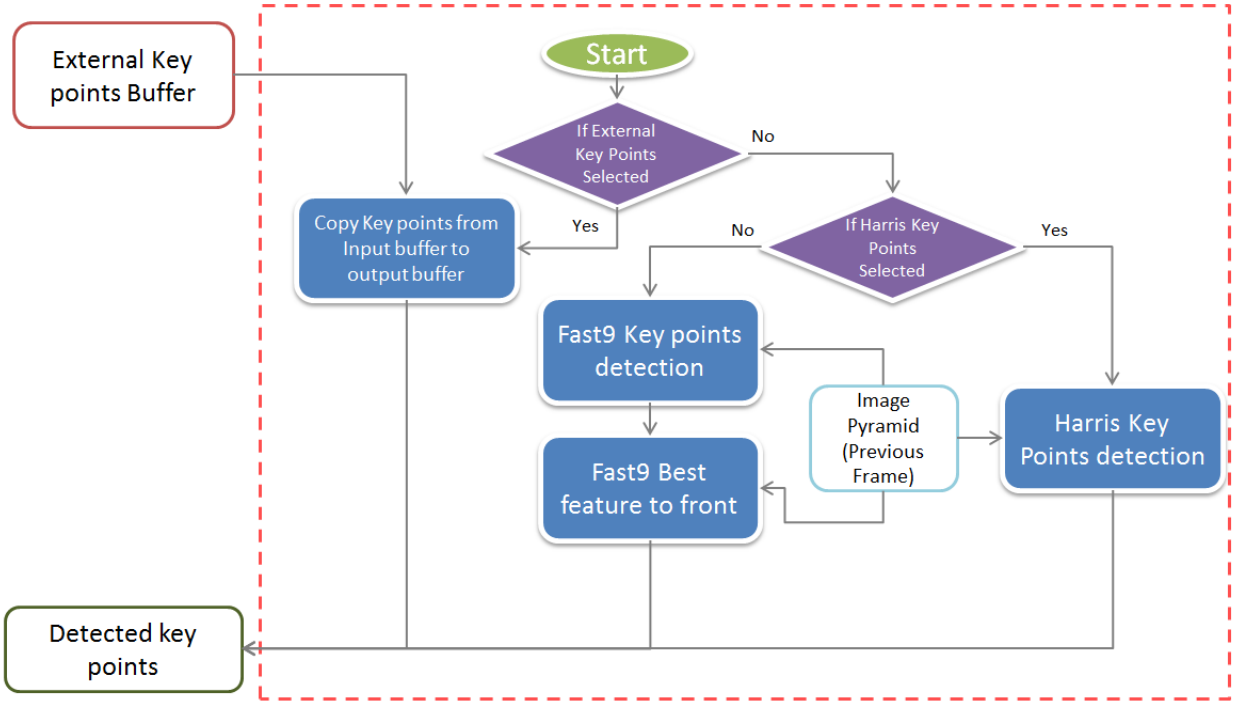
External Key points Buffer
Start
If External Key Points Selected
No
Yes
Copy Key points from Input buffer to output buffer
No
If Harris Key Points Selected
Yes
Fast9 Key points detection
Image Pyramid (Previous Frame)
Harris Key Points detection
Fast9 Best feature to front
Detected key points

**Figure 2 Key Points detection module flow**

# 2. Data Structures

## 2.1 SOF_TI_CreateParams

**|| Description**

This structure defines the creation parameters for SOF algorithm

**|| Fields**

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| visionParams | IVISION_Params | Input | Commom structure for vision modules |
| imWidth | uint16_t | Input | Width in bytes for the input image |
| imHeight | uint16_t | Input | Height in bytes for the input image |
| roiWidth | uint16_t | Input | Processing region width |
| roiHeight | uint16_t | Input | Processing region height |
| startX | uint16_t | Input | X co-ordinate of the first (top-left) pixel in the image. Useful for a ROI processing in a bigger image. Though the information about ROI is supplied via iVISION buf desc but create time also this information is required as this algorithm use the XY value as some create time initialization. It is for base level only. |
| startY | uint16_t | Input | Y co-ordinate of the first (top-left) pixel in the image. Useful for a ROI processing in a bigger image. Though the information about ROI is supplied via iVISION buf desc but create time also this information is required as this algorithm use the XY value as some create time initialization. It is for base level only |
| numLevels | uint8_t | input | Total number of pyramid levels including base level (1 means only base resolution) |
| keyPointDetectMethod | uint8_t | input | The key points in the image can be detetcted can be selected using this parameter |
| keyPointDetectInterval | uint8_t | input | key point detection interval in number of frames. Setting to zero detects the key points in each frame |

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| maxNumKeyPoints | uint16_t | input | Maximum number of frames that need to be detected by the algorithm |
| fast9Threshold | uint16_t | Input | Threshold on difference between intensity of the central pixel and pixels of a circle around this pixel for FAST9 corner detect applet. |
| scoreMethod | uint8_t | Input | Method for FAST9 Score – this score is used for best feature to front. ORB_TI_THRESH_METHOD and ORB_TI_SAD_METHOD are supported values |
| harrisScaling | uint16_t | Input | Scale foactor used in Harris score computation |
| harrisScoreMethod | uint8_t | Input | Method to use for Harris Score calculation. Refer tos HARRIS_CORNER_DETECTION_32_TI_HarrisScoreMethod for valid values defined in "iHarrisCornerDetection32_ti.h" file |
| harrisWindowSize | uint8_t | Input | Window size to be used for harris score calculation. Considers a harrisWindowSize x harrisWindowSize neighborhood to calculate Harris Score. Kindly refer to HARRIS_CORNER_DETECTION_32_TI_HarrisWindowSize for valid values in "iHarrisCornerDetection32_ti.h" file |
| suppressionMethod | uint8_t | Input | Suppression method to be used for non maximum suppression. Kindly refer to HARRIS_CORNER_DETECTION_32_TI_SuppressionMethod for valid values in "iHarrisCornerDetection32_ti.h" file |
| trackOffsetMethod | uint8_t | Input | The method to be used for the output of track coordinates. Kindly refer to SOF_TrackOffsetMethod enum for valid values |
| nmsThreshold | uint32_t | Input | NMS threshold for harris corner detection applet |
| maxItersLK[PYRAMID_LK_TRACKER_TI_MAXLEVELS] | uint16_t | input | Maximum number of iterations for the iterative loop of pyramid LK tracker applet. This value can be set individually for each level. |

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| minErrValue[PYRAMID_LK_TRACKER_TI_MAXLEVELS] | uint16_t | Input | Minimum flow vector difference value at any iteration of a given pyramid level. This input is represented using Q10 format. If the motion detected for a given point is less than or equal to this threshold, then it is considered as negligible motion and thereby invokes exit from the iterative loop of pyramid LK tracker. This value can be set individually for each level. |
| searchRange[PYRAMID_LK_TRACKER_TI_MAXLEVELS]; | uint8_t | Input | Search range in pixel for each level |

### 2.1.1.1 SOF_TI_InArgs

**‖ Description**

This structure contains all the input parameters which are given to SOF algorithm at each process call.

.

**‖ Fields**

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| iVisionInArgs | IVISION_InArgs | Input | Common inArgs for all ivison based modules |
| numCorners | int32_t | Input | Number of valid corners that need to be tracked. This parameter will be used only when SOF_KeyPointDetectMethod is SOF_EXTERNAL_KEY_POINTS. |
| trackErrThr | uint32_t | Input | Estimated error of each key point will be compared against this threshold. If the error is below then  this threshold then the key will be considred for tracking in next frame |
| trackMinFlowQ4 | uint32_t | Input | Minimum flow absolute (abs(vx) + abs(vy)) flow vector of key point that needs to be considered for tracking in next frame |
| trackNmsWinSize | uint32_t | Input | Number of neighbouring pixel (in each direction) to be considred for non maximum suppression while considered key poits to track in next frame. |
| reservered0 | uint8_t | Input | This reserved parameter needs to be set to 0 for optimal performance. |
| staticFrame | uint8_t | Input | A value of 1 Indicates if the current frame is static with respect to previous frame.  A value of 0 indicates a non static frame |

### 2.1.1.2 SOF_TI_OutArgs

**‖ Description**

This structure contains all the parameters which are given as an output by SOF algorithm.

**‖ Fields**

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| `iVisionOutArgs` | `IVISION_Params` | Output | Commom structure for outArgs ivision modules |
| `numCorners` | `int32_t` | Output | Total number of Key points (corners) that is beignt tracked in this frame |

### 2.1.1.3 ISOF_InBufOrder

**Description**

User provides most of the information through buffer descriptor during process call. This enum defines the purpose of the input buffer.

**Fields**

| Enumeration | Description |
|---|---|
| `SOF_TI_BUFDESC_IN_CURR_IMAGEBUFFER` | This buffer descriptor provides the actual image data required by algorithm. |
| `SOF_TI_BUFDESC_IN_KEY_POINTS_BUFFER` | Input key points buffer. This buffer will only be used when the SOF_KeyPointDetectMethod is SOF_EXTERNAL_KEY_POINTS |
| `SOF_TI_BUFDESC_IN_TOTAL` | This indicates total number of input buffers expected. |

### 2.1.1.4 ISOF_OutBufOrder

**Description**

User provides most of the information through buffer descriptor during process call. This enum defines the purpose of the output buffer.

**Fields**

| Enumeration | Description |
|---|---|

| Enumeration | Description |
|---|---|
| SOF_TI_BUFDESC_OUT_KEY_POINTS_BUFFER | This buffer descriptor should point to a plane capable of holding XY list of tracked Key points current location in the current frame. so the size of this buffer is maxNumKeyPoints*(4) bytes. This buffer holds the output of the LK tracker modules without any pruning. This buffer is mostly meant to be used for the cases when we provide input key points externally. It is also important to note that this buffer may not be in sync with the trackInfo buffer because trackInfo buffer is populated after pruning the output of LK tracker based of errEstimate or to limit the number of output tracks |
| SOF_TI_BUFDESC_OUT_ERR_ESTIMATE_BUFFER | This buffer descriptor should point to a plane capable of holding esitmated error measure (SAD based) for each key point that is being tracked. so the size of this buffer is maxNumKeyPoints*(2) bytes. This buffer holds the errEstimate output of the LK tracker modules before any pruing. This buffer will be in sync with the SOF_TI_BUFDESC_OUT_KEY_POINTS_BUFFER. This buffer is mostly meant to be used for the cases when we provide input key points externally. It is also important to note that this buffer may not be in sync with the trackInfo buffer because trackInfo buffer is populated after pruning the output of LK tracker based of errEstimate or to limit the number of output tracks |
| SOF_TI_BUFDESC_OUT_TRACK_INFO_BUFFER | This buffer descriptor should point to a plane capable of holding tracking infomation for each key point that is being tracked across MAX_NUM_FRAMES_TO_TRACK frames. so the size of this buffer is maxNumKeyPoints*sizeof(strackInfo) bytes. The format in which trach info coordinates will be generated will depend on trackOffsetMethod which is part of SOF createparams |
| SOF_TI_BUFDESC_OUT_TOTAL | This indicates total number of output buffers expected. |

### 2.1.1.5 strackInfo

**‖ Description**

This structure contains tracking information for each key points.

**‖ Fields**

| Field | Data Type | Input/ Output | Description |
|---|---|---|---|
| age | uint16_t | Output | Age of the current key points that is being tracked. If the age is zero then the key point is not valid. After 65535 (2^16 - 1) age will be set to (65535 - MAX_NUM_FRAMES_TO_TRACK) |
| x | uint16_t | Output | Array of horizontal co-oridinates in Q4 format for MAX_NUM_FRAMES_TO_TRACK frames |
| y | uint16_t | Output | Array of vertical co-oridinates in Q4 format for MAX_NUM_FRAMES_TO_TRACK frames |

## 2.1.1.6 ISOF_ErrorType

**‖ Description**

Error code returned by the SOF algorithm.

**‖ Fields**

| Enums | Description |
|---|---|
| ISOF_ERRORTYPE_MAXLEVELS_EXCEEDED | The number of levels request by user are more than supported by SOF algorithm |
| ISOF_ERRORTYPE_INVALID_START_XY | The startX and Y are beyond the image boundary |
| ISOF_ERRORTYPE_INVALID_IMAGE_DIMS | Image dimensions are beyond the supported |
| ISOF_ERRORTYPE_IMGPYRAMID_CREATE_FAIL | Failure while creating image Pyramid |
| ISOF_ERRORTYPE_FAST9_CORNER_CREATE_FAIL | Failure while creating fast9 corner detector |
| ISOF_ERRORTYPE_FAST9_BFFT_CREATE_FAIL | Failure while creating fast9 best feature to front |
| ISOF_ERRORTYPE_YUV_PADDING_CREATE_FAIL | Failure while creating YUV Padding applet |
| ISOF_ERRORTYPE_HARRIS_KD_CREATE_FAIL | Failure while creating Harris corner detection applet |

| Enums | Description |
|---|---|
| ISOF_ERRORTYPE_NOT_VALID_KD_METHOD | Key Point detecttion method requisted user is not supported |
| ISOF_ERRORTYPE_LK_TRACKER_CREATE_FAIL | Failure while creating Pyramidal LK tracker applet |
| ISOF_ERRORTYPE_INSUFF_MEM_FOR_IO_BUFFS | Internal Memory requested by the algorithm is not sufficient for the given configuration |

## 2.1.1.7 SOF_KeyPointDetectMethod

**|| Description**

Key Points detection method supported by SOF algorithm.

**|| Fields**

| Enums | Description |
|---|---|
| SOF_EXTERNAL_KEY_POINTS | Key points are provided by the user through a input buffer |
| SOF_HARRIS_KEY_POINTS | Harris key point detetctor method will be used by the algorithm |
| SOF_FAST9_KEY_POINTS | Fast9 key point detetctor method along with best feature to front will be used by the algorithm |

## 2.1.1.8 SOF_TrackOffsetMethod

**|| Description**

This enum defines the methods, which can be used for different methods in which tracks can be formed.

**|| Fields**

| Enums | Description |
|---|---|
| SOF_TRACK_OFFSET_METHOD_IMAGE_ORIGIN | Tracks coordinates will be with respect to the image origin |
| SOF_TRACK_OFFSET_METHOD_ROI_ORIGIN | Tracks coordinates will be with respect to the ROI origin |

## 2.1.1.9 Macros

**‖ Description**

Macros used by SOF algorithm.

**‖ Fields**

| Macros | Description |
|---|---|
| `SOF_TI_MAXLEVELS` | Max levels allowed by SOF |
| `SOF_TI_MAXWIDTH` | Max Width allowed by SOF |
| `SOF_TI_MAXHEIGHT` | Max Height allowed by SOF |
| `MAX_NUM_FRAMES_TO_TRACK` | Number of frames for which the tracking information needs to be maintained by the algorithm. This value shall be power of 2 |

# Appendix A: Compatibility Information

This section mentions the compatibility table with respect to previous release

| Compatibility Parameters Settings (To achieve 1.11 compatibility) | | | | |
|---|---|---|---|---|
| Applet/algo name | Parameter type | Parameter name | | |
| | | | | |
| | | | | |
| | | | | |

Previous release 1.11 is compatible with 1.12 at this algrotihm level.

# IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components, which TI has specifically designated as military grade or “enhanced plastic”, are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

| **Products** | | **Applications** | |
|---|---|---|---|
| Audio | www.ti.com/audio | Automotive & Transportation | www.ti.com/automotive |
| Amplifiers | amplifier.ti.com | Communications & Telecom | www.ti.com/communications |
| Data Converters | dataconverter.ti.com | Computers & Peripherals | www.ti.com/computers |
| DLP® Products | www.dlp.com | Consumer Electronics | www.ti.com/consumer-apps |
| DSP | dsp.ti.com | Energy and Lighting | www.ti.com/energyapps |
| Clocks and Timers | www.ti.com/clocks | Industrial | www.ti.com/industrial |
| Interface | interface.ti.com | Medical | www.ti.com/medical |
| Logic | logic.ti.com | Security | www.ti.com/security |
| Power Mgmt | power.ti.com | Space, Avionics & Defense | www.ti.com/space-avionics-defense |
| Microcontrollers | microcontroller.ti.com | Video & Imaging | www.ti.com/video |
| RFID | www.ti-rfid.com | | |
| OMAP Applications Processors | www.ti.com/omap | **TI E2E Community** | e2e.ti.com |
| Wireless Connectivity | www.ti.com/wirelessconnectivity | | |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright©
2013, Texas Instruments Incorporated