ANNEXURE III

Software Program for DSP

```
#ifndef DSP281x_GLOBALPROTOTYPES_H
#define DSP281x_GLOBALPROTOTYPES_H


#ifdef __cplusplus
extern "C" {
#endif


#include "IQmathLib.h"                    //

/*---- shared global function prototypes ---------------------------------*/
extern void InitAdc(void);
extern void InitPeripherals(void);
extern void InitECan(void);
extern void InitEv(void);
extern void InitGpio(void);
extern void InitMcbsp(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void EnableInterrupts(void);
extern void InitSci(void);
extern void InitSpi(void);
extern void InitSysCtrl(void);
extern void InitXintf(void);
extern void InitXIntrupt(void);
extern void InitPll(Uint16 val);
extern void InitPeripheralClocks(void);
extern void InitADCOffset(void);
extern void max_value(void);
extern void sector(void);
extern _iq19 vr,vy,vb,VoltageR,VoltageY,VoltageB, maxV;
extern Uint16 scounter,StartFlg;
extern int16 ADCOffset[3];
extern int16 max[3],min[3];

#include "DSP281x_Device.h"     // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"   // DSP281x Examples Include File


// Prototype statements for functions found within this file.
```

```
// Global variables used in this example


// These are defined by the linker (see F2812.cmd)
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;

void main(void)
{

  //InitGpio();
// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP281x_SysCtrl.c file.
  InitSysCtrl();
// Step 2. Clear all interrupts and initialize PIE vector table:
// Disable CPU interrupts
  DINT;

// Initialize PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP281x_PieCtrl.c file.
  InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:
  IER = 0x0000;
  IFR = 0x0000;

// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP281x_DefaultIsr.c.
// This function is found in DSP281x_PieVect.c.
  InitPieVectTable();

// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.
  EALLOW;  // This is needed to write to EALLOW protected registers
  EDIS;   // This is needed to disable write to EALLOW protected registers


// Step 3. Initalize GPIO:
  InitGpio();
```

```
 // Step 4. Initialize all the Device Peripherals:
// This function is found in DSP281x_InitPeripherals.c
  InitPeripherals();

// Step 5. User specific code, enable interrupts:
  EALLOW;  // This is needed to write to EALLOW protected registers
  //PieVectTable.T2PINT = &eva_timer2_isr;
  EDIS;   // This is needed to disable write to EALLOW protected registers


// Copy time critical code and Flash setup code to RAM
// This includes the following functions:  InitFlash();
// The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
// symbols are created by the linker. Refer to the F2812.cmd file.
  MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);

// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM
  InitFlash();

   // Step 6. IDLE loop. Just sit and loop forever:


while(1)
  {

/*************************************************************************
*****/
               // Enable Capture Interrupt

       // Enable PIE group 3 interrupt 5 for CAP1
               PieCtrlRegs.PIEIER3.all = M_INT5;
  // Enable CPU INT3 for CAP1
               IER |= M_INT3;

/*************************************************************************
*******/

  // Enable global Interrupts and higher priority real-time debug events:

               EINT;  // Enable Global interrupt INTM
               //ERTM;  // Enable Global realtime interrupt DBGM

               while (StartFlg <= 4)
               {
```

```
                InitADCOffset();
                }

                ReadADC();
                max_value();
                sector();


                }

        }

void InitADCOffset(void)
{

  while(1)
                {
                if (StartFlg >= 3)
                        {

                        AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;
                        while (AdcRegs.ADCASEQSR.bit.SEQ_CNTR ||
AdcRegs.ADCST.bit.SEQ1_BSY);   //Wait for completion of conv seq
                        if((AdcRegs.ADCRESULT0 >> 4) > max[0])
                                max[0] = (AdcRegs.ADCRESULT0 >> 4);
                        if((AdcRegs.ADCRESULT0 >> 4) < min[0])
                                min[0] = (AdcRegs.ADCRESULT0 >> 4);

                        if((AdcRegs.ADCRESULT1 >> 4) > max[1])
                                max[1] = (AdcRegs.ADCRESULT1 >> 4);
                        if((AdcRegs.ADCRESULT1 >> 4) < min[1])
                                min[1] = (AdcRegs.ADCRESULT1 >> 4);

                        if((AdcRegs.ADCRESULT2 >> 4) > max[2])
                                max[2] = (AdcRegs.ADCRESULT2 >> 4);
                        if((AdcRegs.ADCRESULT2 >> 4) < min[2])
                                min[2] = (AdcRegs.ADCRESULT2 >> 4);

                        }
                if(StartFlg >= 4)
                        break;
                }
  for(scounter = 0; scounter < 3; scounter++)
                {
                ADCOffset[scounter] = ( ( max[scounter] + min[scounter] ) >> 1 );//16
subtracted for offset comp
```

```
            }                                                    //2116
    scounter = 0;
}


void ReadADC(void)
{
        vr = /*_IQ19*/( (AdcRegs.ADCRESULT0 >> 4) - (ADCOffset[0]) );
        vr = vr << 19;//_IQ19(vr);
        vy = /*_IQ19*/( (AdcRegs.ADCRESULT1 >> 4) - (ADCOffset[1]) );
        vy = vy << 19;//_IQ19(vy);
        vb = /*_IQ19*/( (AdcRegs.ADCRESULT2 >> 4) - (ADCOffset[2]) );
        vb = vb << 19;//_IQ19(vb);

}
        void max_value(void)
{

        VoltageR = vr;
        VoltageY = vy;
        VoltageB = vb;
        VoltageR = _IQ19abs(VoltageR);
        VoltageY = _IQ19abs(VoltageY);
        VoltageB = _IQ19abs(VoltageB);


}

  void sector(void)
 {



            if(VoltageR > VoltageY && VoltageR > VoltageB && vr > 0)
            {

            //      sector1();
            }

            else if(VoltageR > VoltageY && VoltageR > VoltageB && vr < 0)
            {

            //      sector4();
            }
```

```
            else if(VoltageY > VoltageB && VoltageY > VoltageR && vy > 0)
            {
                // sector3();
            }

            else if(VoltageY > VoltageB && VoltageY > VoltageR && vy < 0)
            {

                //sector6();
            }

            else if(VoltageB > VoltageR && VoltageB > VoltageY && vb > 0)
            {

                //sector5();
            }

            else if(VoltageB > VoltageR && VoltageB > VoltageY && vb < 0)
            {

                //sector2();
            }

 }

// Initialize ADC Peripheral To default State:
   //InitAdc();
        // To powerup the ADC the ADCENCLK bit should be set first to enable
   // clocks, followed by powering up the bandgap and reference circuitry.
   // After a 5ms delay the rest of the ADC can be powered up. After ADC
   // powerup, another 20us delay is required before performing the first
   // ADC conversion. Please note that for the delay function below to
   // operate correctly the CPU_CLOCK_SPEED define statement in the
   // DSP28_Examples.h file must contain the correct CPU clock period in
   // nanoseconds. For example:

        AdcRegs.ADCTRL3.bit.ADCBGRFDN = 0x3;       // Power up
bandgap/reference circuitry
        for(del = 0; del < 65000; del++)asm(" RPT #7 || NOP");
        //DELAY_US(ADC_usDELAY);              // Delay before powering up rest of
ADC
        AdcRegs.ADCTRL3.bit.ADCPWDN = 1;            // Power up rest of ADC
        for(del = 0; del < 2500; del++)asm(" RPT #7 || NOP");
        //DELAY_US(ADC_usDELAY2);             // Delay after powering up AD
```

```
        AdcRegs.ADCTRL1.bit.ACQ_PS = ADC_SHCLK;  // Sequential mode: Sample
rate   = 1/[(2+ACQ_PS)*ADC clock in ns]
                                                 //                = 1/(3*40ns) =8.3MHz
                                                 // If Simultaneous mode enabled: Sample
rate = 1/[(3+ACQ_PS)*ADC clock in ns]
  AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC_CKPS;
  AdcRegs.ADCTRL3.bit.SMODE_SEL = 0;         //Sequential sampling mode
  AdcRegs.ADCTRL1.bit.SEQ_CASC = 0;        // 1  Cascaded mode
  AdcRegs.ADCTRL1.bit.CONT_RUN = 1;       // Setup start/stop mode
  AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 0;            //Clear pending SOC trigger

  //AdcRegs.ADCTRL1.bit.SEQ_OVRD = 1;       // Enable Sequencer override feature
  AdcRegs.ADCCHSELSEQ1.all = 0x0210;        // Initialize all ADC channel selects
  //AdcRegs.ADCCHSELSEQ2.all = 0x0BA98;//0x7654;//0x0BA98
 // AdcRegs.ADCCHSELSEQ3.all = 0x0FEDC;//0x0BA98;
  //AdcRegs.ADCCHSELSEQ4.all = 0x0FEDC;
  AdcRegs.ADCMAXCONV.bit.MAX_CONV1 = 0x02;//0x0F;  // convert and store in
16 results registers
```

ANALOG DEVICES

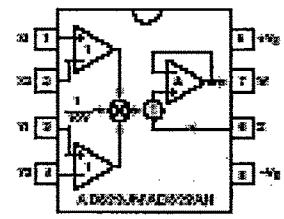
Low Cost Analog Multiplier

AD633

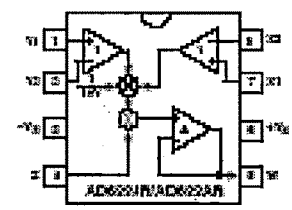
FEATURES
4-Quadrant Multiplication
Low Cost 8-Lead Package
Complete—No External Components Required
Laser-Trimmed Accuracy and Stability
Total Error within 2% of FS
Differential High Impedance X and Y Inputs
High Impedance Unity-Gain Summing Input
Laser-Trimmed 10 V Scaling Reference

APPLICATIONS
Multiplication, Division, Squaring
Modulation/Demodulation, Phase Detection
Voltage Controlled Amplifiers/Attenuators/Filters

CONNECTION DIAGRAMS
8-Lead Plastic DIP (N) Package



8-Lead Plastic SOIC (RN-8) Package



PRODUCT DESCRIPTION
The AD633 is a functionally complete, four-quadrant, analog multiplier. It includes high impedance, differential X and Y inputs and a high impedance summing input (Z). The low impedance output voltage is a nominal 10 V full scale provided by a buried Zener. The AD633 is the first product to offer these features in modestly priced 8-lead plastic DIP and SOIC packages.

The AD633 is laser calibrated to a guaranteed total accuracy of 2% of full scale. Nonlinearity for the Y input is typically less than 0.1% and noise referred to the output is typically less than 100 μV rms in a 10 Hz to 10 kHz bandwidth. A 1 MHz bandwidth, 20 V/μs slew rate, and the ability to drive capacitive loads make the AD633 useful in a wide variety of applications where simplicity and cost are key concerns.

The AD633's versatility is not compromised by its simplicity. The Z-input provides access to the output buffer amplifier, enabling the user to sum the outputs of two or more multipliers, increase the multiplier gain, convert the output voltage to a current, and configure a variety of applications.

The AD633 is available in an 8-lead plastic DIP package (N) and 8-lead SOIC (R). It is specified to operate over the 0°C to 70°C commercial temperature range (J Grade) or the –40°C to +85°C industrial temperature range (A Grade).

PRODUCT HIGHLIGHTS
1. The AD633 is a complete four-quadrant multiplier offered in low cost 8-lead plastic packages. The result is a product that is cost effective and easy to apply.
2. No external components or expensive user calibration are required to apply the AD633.
3. Monolithic construction and laser calibration make the device stable and reliable.
4. High (10 MΩ) input resistances make signal source loading negligible.
5. Power supply voltages can range from ±8 V to ±18 V. The internal scaling voltage is generated by a stable Zener diode; multiplier accuracy is essentially supply insensitive.

REV. E

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 www.analog.com
Fax: 781/326-8703

AD633—SPECIFICATIONS (T_A = 25°C, V_S = ±15 V, $R_L \geq 2$ kΩ)

Model		AD633J, AD633A			
TRANSFER FUNCTION		$W = \frac{(X_1 - X_2)(Y_1 - Y_2)}{10\ V} + Z$			
Parameter	Conditions	Min	Typ	Max	Unit
MULTIPLIER PERFORMANCE					
Total Error	−10 V ≤ X, Y ≤ +10 V		±1	±2	% Full Scale
T_{MIN} to T_{MAX}			±3		% Full Scale
Scale Voltage Error	SF = 10.00 V Nominal		±0.25%		% Full Scale
Supply Rejection	V_S = ±14 V to ±16 V		±0.01		% Full Scale
Nonlinearity, X	X = ±10 V, Y = +10 V		±0.4	±1	% Full Scale
Nonlinearity, Y	Y = ±10 V, X = +10 V		±0.1	±0.4	% Full Scale
X Feedthrough	Y Nulled, X = ±10 V		±0.3	±1	% Full Scale
Y Feedthrough	X Nulled, Y = ±10 V		±0.1	±0.4	% Full Scale
Output Offset Voltage			±5	±50	mV
DYNAMICS					
Small Signal BW	V_O = 0.1 V rms		1		MHz
Slew Rate	V_O = 20 V p-p		20		V/μs
Settling Time to 1%	ΔV_O = 20 V		2		μs
OUTPUT NOISE					
Spectral Density			0.8		μV/√Hz
Wideband Noise	f = 10 Hz to 5 MHz		1		mV rms
	f = 10 Hz to 10 kHz		90		μV rms
OUTPUT					
Output Voltage Swing		±11			V
Short Circuit Current	R_L = 0 Ω		30	40	mA
INPUT AMPLIFIERS					
Signal Voltage Range	Differential	±10			V
	Common Mode	±10			V
Offset Voltage X, Y			±5	±30	mV
CMRR X, Y	V_{CM} = ±10 V, f = 50 Hz	60	80		dB
Bias Current X, Y, Z			0.8	2.0	μA
Differential Resistance			10		MΩ
POWER SUPPLY					
Supply Voltages					
Rated Performance			±15		V
Operating Range		±8		±18	V
Supply Current	Quiescent		4	6	mA

Specifications shown in boldface are tested on all production units at electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed. Although only those shown in boldface are tested on all production units.
Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS[1]

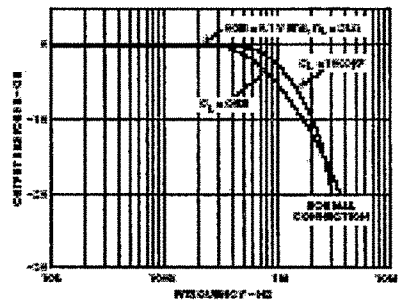
Supply Voltage	±18 V
Internal Power Dissipation[2]	500 mW
Input Voltages[3]	±18 V
Output Short Circuit Duration	Indefinite
Storage Temperature Range	−65°C to +150°C
Operating Temperature Range	
AD633J	0°C to 70°C
AD633A	−40°C to +85°C
Lead Temperature Range (Soldering 60 sec)	300°C
ESD Rating	1000 V

NOTES
[1]Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied.
[2]8-Lead Plastic DIP Package: θ_{JA} = 90°C/W; 8-Lead Small Outline Package: θ_{JA} = 155°C/W.
[3]For supply voltages less than ±18 V, the absolute maximum input voltage is equal to the supply voltage.

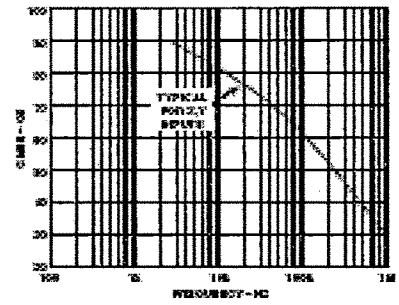
ORDERING GUIDE

Model	Temperature Range	Package Description	Package Option
AD633AN	−40°C to +85°C	Plastic DIP	N-8
AD633AR	−40°C to +85°C	Plastic SOIC	RN-8
AD633AR-REEL	−40°C to +85°C	13" Tape and Reel	RN-8
AD633AR-REEL7	−40°C to +85°C	7" Tape and Reel	RN-8
AD633JN	0°C to 70°C	Plastic DIP	N-8
AD633JR	0°C to 70°C	Plastic SOIC	RN-8
AD633JR-REEL	0°C to 70°C	13" Tape and Reel	RN-8
AD633JR-REEL7	0°C to 70°C	7" Tape and Reel	RN-8

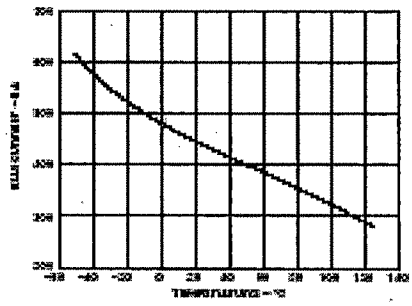
Typical Performance Characteristics—AD633



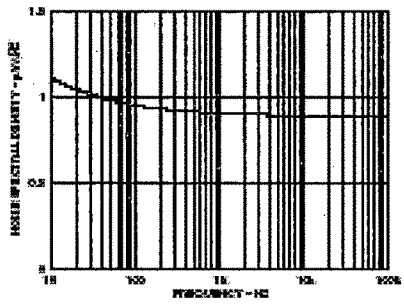
TPC 1. Frequency Response



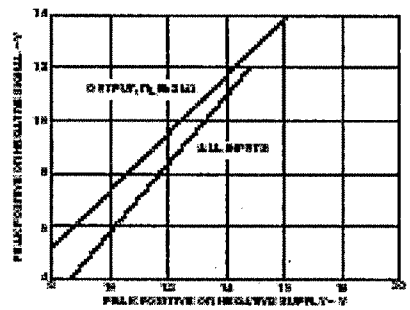
TPC 4. CMRR vs. Frequency



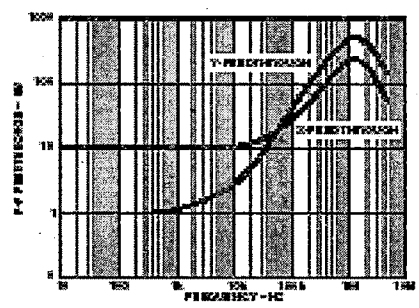
TPC 2. Input Bias Current vs. Temperature (X, Y, or Z Inputs)



TPC 5. Noise Spectral Density vs. Frequency



TPC 3. Input and Output Signal Ranges vs. Supply Voltages



TPC 6. AC Feedthrough vs. Frequency

FUNCTIONAL DESCRIPTION

The AD633 is a low cost multiplier comprising a translinear core, a buried Zener reference, and a unity gain connected output amplifier with an accessible summing node. Figure 1 shows the functional block diagram. The differential X and Y inputs are converted to differential currents by voltage-to-current converters. The product of these currents is generated by the multiplying core. A buried Zener reference provides an overall scale factor of 10 V. The sum of $(X_1 - X_2)(Y_1 - Y_2)/10 + Z$ is then applied to the output amplifier. The amplifier summing node Z allows the user to add two or more multiplier outputs, convert the output voltage to a current, and configure various analog computational functions.

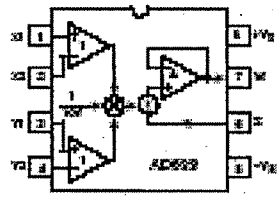


Figure 1. Functional Block Diagram (AD633JN Pinout Shown)

Inspection of the block diagram shows the overall transfer function to be:

$$W = \frac{(X_1 - X_2)(Y_1 - Y_2)}{10\ V} + Z \qquad (1)$$

ERROR SOURCES

Multiplier errors consist primarily of input and output offsets, scale factor error, and nonlinearity in the multiplying core. The input and output offsets can be eliminated by using the optional trim of Figure 2. This scheme reduces the net error to scale factor errors (gain error) and an irreducible nonlinearity component in the multiplying core. The X and Y nonlinearities are typically 0.4% and 0.1% of full scale, respectively. Scale factor error is typically 0.25% of full scale. The high impedance Z input should always be referenced to the ground point of the driven system, particularly if this is remote. Likewise, the differential X and Y inputs should be referenced to their respective grounds to realize the full accuracy of the AD633.

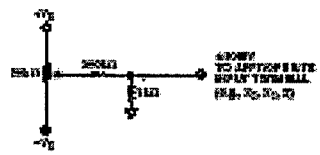


Figure 2. Optional Offset Trim Configuration

APPLICATIONS

The AD633 is well suited for such applications as modulation and demodulation, automatic gain control, power measurement, voltage controlled amplifiers, and frequency doublers. Note that these applications show the pin connections for the AD633JN pinout (8-lead DIP), which differ from the AD633JR pinout (8-lead SOIC).

Multiplier Connections

Figure 3 shows the basic connections for multiplication. The X and Y inputs will normally have their negative nodes grounded, but they are fully differential, and in many applications the grounded inputs may be reversed (to facilitate interfacing with signals of a particular polarity while achieving some desired output polarity) or both may be driven.

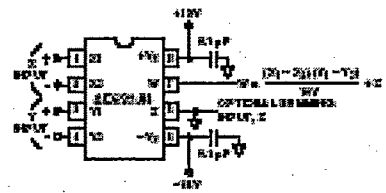


Figure 3. Basic Multiplier Connections

Squaring and Frequency Doubling

As Figure 4 shows, squaring of an input signal, E, is achieved simply by connecting the X and Y inputs in parallel to produce an output of $E^2/10$ V. The input may have either polarity, but the output will be positive. However, the output polarity may be reversed by interchanging the X or Y inputs. The Z input may be used to add a further signal to the output.

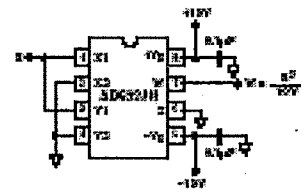


Figure 4. Connections for Squaring

When the input is a sine wave $E \sin \omega t$, this squarer behaves as a frequency doubler, since

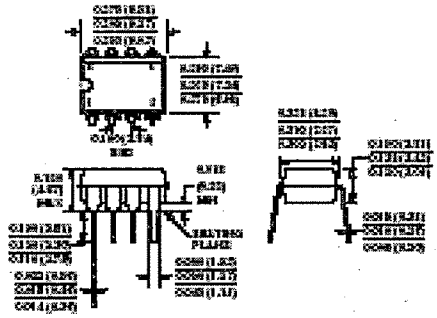
$$\frac{(E \sin \omega t)^2}{10\ V} = \frac{E^2}{20V}(1 - \cos 2\omega t) \qquad (2)$$

Equation 2 shows a dc term at the output that will vary strongly with the amplitude of the input, E. This can be avoided using the connections shown in Figure 5, where an RC network is used to generate two signals whose product has no dc term. It uses the identity:

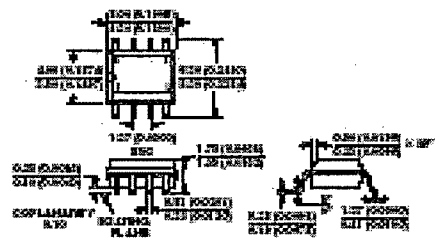
$$\cos\theta \sin\theta = \frac{1}{2}(\sin 2\theta) \qquad (3)$$

OUTLINE DIMENSIONS

8-Lead Plastic Dual-in-Line Package [PDIP]
(N-8)
Dimensions shown in inches and (millimeters)



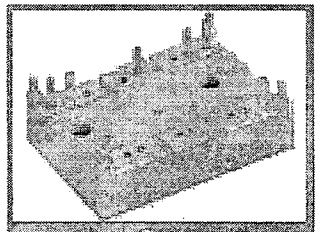
8-Lead Standard Small Outline Package [SOIC]
Narrow Body
(RN-8)
Dimensions shown in millimeters and (inches)



Revision History

Location	Page
10/02—Data Sheet changed from REV. D to REV. E.	
Edits to title of 8-Lead Plastic SOIC Package (RN-8)	1
Edits to ORDERING GUIDE	2
Change to Figure 13	7
Updated OUTLINE DIMENSIONS	8

SK 60 GM 123



IGBT Module

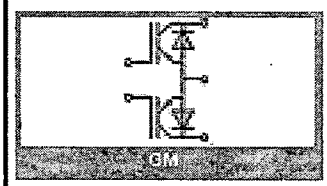
SK 60 GM 123
Preliminary Data

Features

- Compact design
- One screw mounting
- Heat transfer and isolation through direct copper bonded aluminium oxide ceramic (DBC)
- High short circuit capability
- Low tail current with low temperature dependance

Typical Applications

- Switching (not for linear use)
- Inverter
- Switched mode power supplies
- UPS

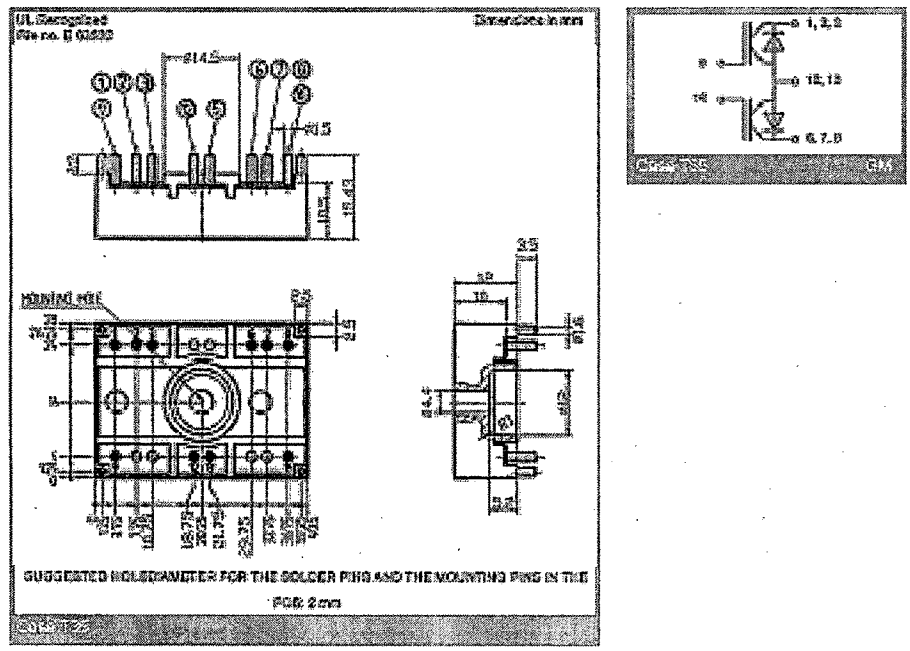


Absolute Maximum Ratings T_s = 25 °C, unless otherwise specified

Symbol	Conditions	Values	Units
IGBT			
V_{CES}		1200	V
V_{GES}		± 20	V
I_C	T_s = 25 (80) °C;	60 (40)	A
I_{CRM}	t_p < 1 ms; T_s = 25 (80) °C;	120 (80)	A
T_j		-40 ... + 150	°C
Inverse / Freewheeling CAL diode			
I_F = - I_C	T_s = 25 (80) °C;	60 (40)	A
I_{FRM} = - I_{CM}	t_p < 1 ms; T_s = 25 (80) °C;	120 (80)	A
T_j		-40 ... + 150	°C
T_{stg}		-40 ... + 125	°C
T_{sol}	Terminals, 10 s	260	°C
V_{isol}	AC 50 Hz, r.m.s. 1 min. / 1 s	2500 / 3000	V

Characteristics T_s = 25 °C, unless otherwise specified

Symbol	Conditions	min.	typ.	max.	Units
IGBT					
$V_{CE(sat)}$	I_C = 50 A; T_j = 25 (125) °C		2,5 (3,1)	3 (3,7)	V
$V_{GE(th)}$	V_{GE} = V_{CE}; I_C = 0,002 A	4,5	5,5	6,5	V
C_{ies}	V_{CE} = 25 V; V_{GE} = 0 V; 1 MHz		3,3		nF
$R_{th(j-s)}$	per IGBT			0,6	K/W
	per module				K/W
	under following conditions:				
$t_{d(on)}$	V_{CC} = 600 V; V_{GE} = ± 15 V		40		ns
t_r	I_C = 50 A; T_j = 125 °C		40		ns
$t_{d(off)}$	R_{Gon} = R_{Goff} = 33 Ω		300		ns
t_f			40		ns
E_{on} + E_{off}	inductive load		12,2		mJ
Inverse / Freewheeling CAL diode					
V_F = V_{EC}	I_F = 50 A; T_j = 25 (125) °C		2 (1,8)	2,5	V
$V_{(TO)}$	T_j = (125) °C		(1)	(1,2)	V
r_T	T_j = (125) °C		(18)	(22)	mΩ
$R_{th(j-s)}$				0,7	K/W
	under following conditions:				
I_{RRM}	I_F = 30 A; V_R = 600 V		14		A
Q_{rr}	di_F/dt = -800 A/µs		5,4		µC
E_{rr}	V_{GE} = 0 V; T_j = 125 °C		2,4		mJ
Mechanical data					
M1	mounting torque			2	Nm
w			23		g
Case	SEMITOP® 2		T 35		



This is an electrostatic discharge sensitive device (ESDS), international standard IEC 60747-1, Chapter IX.

This technical information specifies semiconductor devices but promises no characteristics. No warranty or guarantee expressed or implied is made regarding delivery, performance or suitability.