Touch Sensors

Design considerations for sensor designs for the PE5004 capacitive sensing controller

SERMA PRODUCTIVITY ENGINEERING

Introduction

The PE5004 is a high precision capacitive sensing circuit which uses amplitude modulation for very fast stimulating and reading of capacitive sensor arrays and single sensors.
A number of up to 100 sensors per chip can be selected by user, multiple sensor arrays can be cascaded. All Sensors can be addressed individually making it possible to integrate several different types of buttons, sliders, wheels and touch matrix arrays with one single PE5004.

Capacitive sensor elements are highly sensitive to environmental and noise effects, making an elaborate design of the board layout and firmware inevitable for a successful product.
This application note provides information on how to set up an application with capacitive sensor controls. Hints are given for an optimal board layout, increasing of the usable resolution and the implementation of firmware that fits the requirements of the application.
Furthermore an example is given how to combine multiple sensor arrays with paying particular considerations to the given layout guidelines.

For information on the working principle and complete description of the PE5004 and a guide how to design capacitive sensors please refer to the PE5004 data sheet.

Table of Content

List of Figures

1 Revision History

Version	Date	Changes	Page
Initial Version V1.0	01/2010		

2 Optimizing layout for high SNR

Measuring of the capacitance is done by a highly sensitive current to voltage converter. Therefore all noise sources inducing even smallest amounts of current into the sensing line show the same behavior like changes in capacitance. As the PE5004 features a lock in amplifier most noise not in the frequency band of the stimulating electrodes is filtered and not affecting the circuit. Frequencies near the stimulating one will inevitably cover with the stimulating phase randomly and therefore lead to unwanted noise.

Noise is caused by the application itself, digital communication, all switching events like PWM for LEDs or pumping of voltage converters.
This kind of noise can be reduced by good board layout. Power inputs shall be buffered with a minimum 1uF Capacitor to ground near the pin. Critical noise sources shall be placed as far away as possible from sensors and PE5004. Sensing wires shall be shielded with a non switching potential. Wherever applicable switching frequencies shall be chosen as far away from sensor stimulating frequency of the PE5004 as possible, frequencies exact on higher orders of the stimulating frequency can also cause problems. If crossing of switching signals cannot be avoided, they shall cross in a 90° angle and must not run parallel. Best sensing performance is achieved for symmetric sensing wires. Therefore it is helpful to implement dummy lines on constant potential to assure equal parasitic capacitance.

Noise also is caused by external sources, radiating electric fields and magnetic fields induced in wire loops.
The easiest ways to reduce external noise is to avoid loops, route wires as short and place them as near as possible to the PE5004. There is often no way to place the chip near all electrodes when setting up a large sensor matrix. In this case it is more important to shorten the sensing wires than the stimulating wires as they are less sensitive to noise. Wherever applicable a grounded shielding mesh (10 - 30% fill) shall be placed behind and around sensing plates. A solid shield yields to better noise absorption but also decreases signal strength dramatically by increasing the parasitic capacitance. A large ground plane area also can capture ground noise that leads to poor performance.

The sensed object can be a large source of noise itself, especially a human body acts as antenna picking up radiating fields and directly couples them to the sensor. Because of direct coupling this kind of noise is hard to prevent. Shielding electrodes help protecting non touched sensors from body noise. The influence of coupled noise can be reduced by firmware techniques like spreading the frequency spectrum, filtering and EMI Pulse routines.

3 Reaching high resolution with few sensors

When considering the setup of a slider, wheel or touch screen the resolution is an important criterion. Here it has to be differentiated between real physical resolution (respectively the multi touch resolution), which directly complies to the number of used sensors, and the interpolated resolution, which is deduced from the Sensors 10bit ADC Value.

The number of Sensors is limiting the possible number of multi touch events and their minimum allowed distance to each other. When the space between two sensors is bigger than the nearest distance two simultaneous interacting objects may have (typical the diameter of a finger), there is no possibility of telling if two weak touch points or one big touch is detected. Both would result in the nearby same changing of the fringe field of the surrounding sensors.

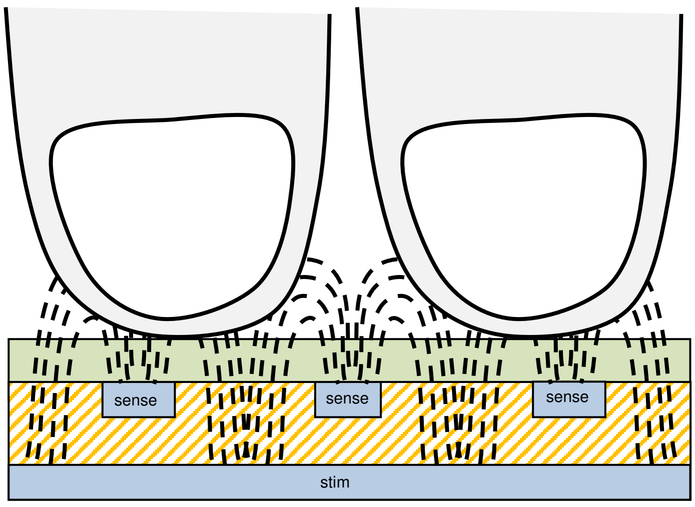

Figure 1 – sensor fringe field

In almost all cases a high multi touch resolution is less important than interpolated resolution and therefore the position of a single touch. Multi touch on screens is commonly used for recognition of multi finger gestures. Here it is important to recognize exact movement of two or more fingers which are separated with enough space to interpolate each finger on its own. At worst case the fingers pinch together, e.g. for a zoom-in. The exact movement of the fingers after they where pinched together is not necessary to know, as the zoom-in recognition is performed already. When they are moving apart this can be easily recognized.
When using the multi touch ability of the PE5004 for implementing multiple buttons, sliders and wheels, the physical separation between them is large enough for not interfering with each other.

The interpolated resolution can be much higher than the number of sensors, as sliding from one sensor to another directly changes the sensor value weight. The possible resolution depends on the Signal to Noise Ratio (SNR) and the ratio between fully activated and untouched sensor. The following figure shows how the measured value of 2 Sensors changes when moving with an object from one sensor to another.

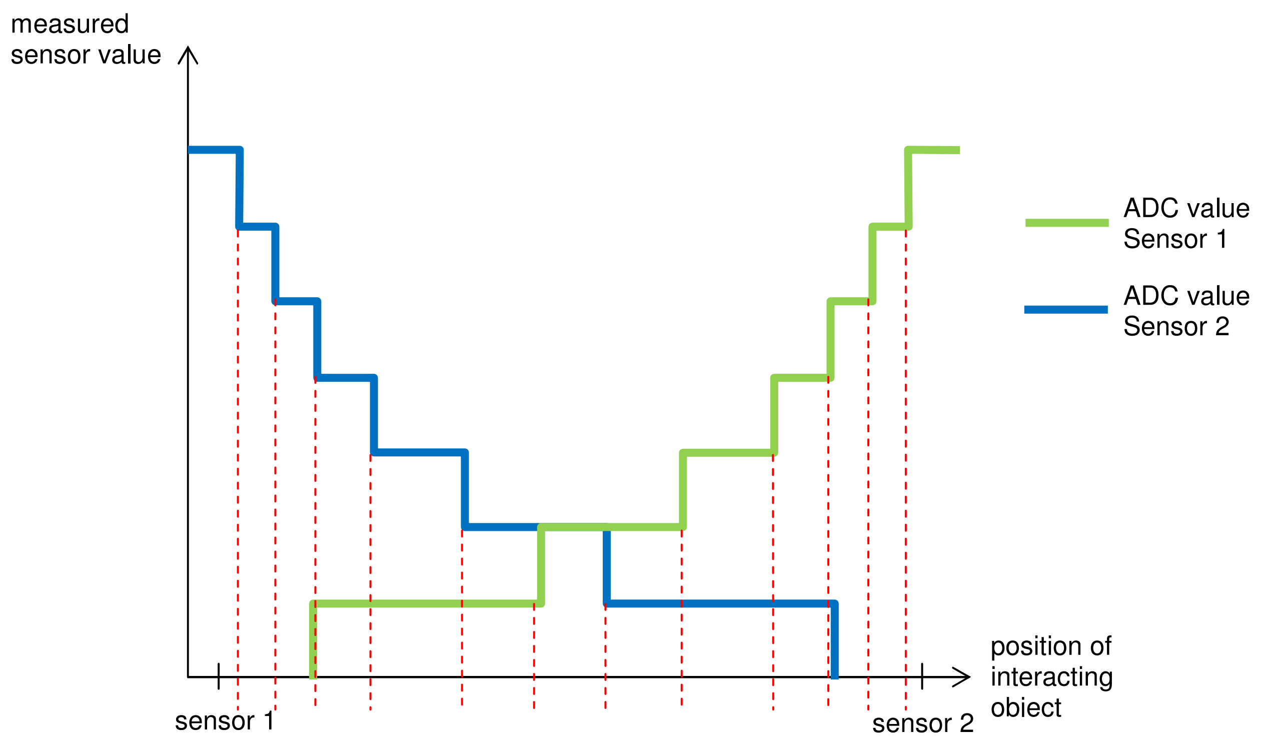

Figure 2 – interpolating for higher resolution

Interpolating the position of one touch can be realized over a simple weighting algorithm:

$$X_{weight} = \frac{\sum_X \sum_Y X \cdot SAR\_DATA_{X,Y}}{\sum_X \sum_Y SAR\_DATA_{X,Y}}; Y_{weight} = \frac{\sum_Y \sum_X Y \cdot SAR\_DATA_{X,Y}}{\sum_Y \sum_X SAR\_DATA_{X,Y}}$$

C Code for algorithm:

```
for(int row =0; row< rowend; row++)
        for(int col =0; col< colend; col++) {
                if (Value_ADC[row][col]>0){
                        vec_x += (col * Value_ADC[row][col]);
                        vec_y += (row * Value_ADC[row][col]);
                        weight += Value_ADC[row][col];
                }
        }
vec_x_weight = (vec_x/weight);
vec_y_weight = (vec_y/weight);
```

Interpolation of more than one touch is more difficult, as touched sensors can overlay each other.
At first the number and absolute position of touches in form of maxima must be determined.
In a second step it is one possibility to adapt the single touch weighting algorithm for each cloud of touch points. To handle the overlay of multiple touch point clouds, the sum of all ADC values must be calculated and divided by the number of touches to get the weight of one cloud. With the knowledge of the single weight it can be determined how large the part of one cloud in a shared sensor fields is. As the weight depends on the size of interacting object and the touch pressure, the interpolation of touch points near each other can only be considered as an average of the real position.

Another method is, after picking all touch points, to look at the slope to all neighbor sensors (generally for a x-y matrix 4 at 90° analysis and 8 at 45°). This can be used to calculate the position on each axis but care has to be taken, as the sensor value is not linear related to the distance of the touching object.

C-Code for 4 direction slope analysis with on-the-fly detection of touch point maxima:

```
for(int col =0; col< colend; col++) my[0][col+1]=Value_ADC[0][col];
for(int row =0; row< rowend; row++){
  mx[row+1][0]=Value_ADC[row][0];
  for(int col =0; col< *iCol; col++) {
    mx[row+1][col+1]=(Value_ADC[row][col+1]-Value_ADC[row][col]);
    my[row+1][col+1]=(Value_ADC[row+1][col]-Value_ADC[row][col]);

    if (mx[row+1][col+1]<0) if (my[row+1][col+1]<0) if (mx[row+1][col]>0) if (my[row][col+1]>0){
      touch_count++;
      touchval[touch_count]=Value_ADC[row][col];
      xtouch[touch_count]=(col+((mx[row+1][col]+mx[row+1][col+1])/(mx[row+1][col]-mx[row+1][col+1])));
      ytouch[touch_count]=(row+((my[row][col+1]+my[row+1][col+1])/(my[row][col+1]-my[row+1][col+1])));
      if (Value_ADC[row][col] > weight) weight=Value_ADC[row][col];
    }
  }
}
```

4 Firmware algorithms for sensor data acquisition

Good board layout and intelligent analog design of the measurement circuit covers most of the unwanted effects which can occur. Nevertheless some disturbances are of physical nature and must be excluded by signal analysis under knowledge of the environmental conditions. In most cases correction algorithms need more than one sample. This leads to higher response time, so these algorithms must be chosen depending on the application and the required accuracy.

4.1 Sensor Baseline

The Sensor Baseline is the converted value of the sensor with no switching or touching event. It depends only on parasitics, the sensor base capacitance and the register configuration of the PE5004. Under difficult environmental conditions it is necessary to frequently re-adjust the Baseline, as the capacity of the sensor changes with humidity and temperature.
The update rate depends on the possible speed of occurring changes. A high baseline update rate assures very quick response to environmental changes, but also leads to short activation pulses, as a long pressure of the button would also be canceled. This can only be avoided by making a baseline update dependent on a non activation. This has to be considered cautiously, as unwanted fast changes can lead to a hang-up of the system. Under normal conditions the baseline update frequency can be chosen much lower than the time constant of a touch or activation would result in.

Making the baseline update only at a simple timer event can lead to missing of a touch which occurred slightly before the update. A good compromise is defining a base line update threshold, watching for changes within the time and updating only for small changes. Activation would change the measured value and prohibit an update. A strong environmental change would not cause a hang-up.

Depending on the sensed object, there is one preferred direction the sensor value would change to while being activated. For example with a human user the fringe field is interrupted and the value lowers. An increasing of the measured value would occur on release of the touch sensor or contamination with water. These are both events which make an instant base line update necessary. Therefore a negative threshold shall be defined, covering the change of measurement value in negative direction (means the direction opposite to the touch which can be also positive).

4.2 Digital Filtering

For high precision applications a continuous filtering algorithm is recommended:

$$Filtered\_Value_{new} = \frac{Filtered\_Value_{old} \cdot (n-1) + Measured\_Value_{ADC}}{n}$$

Where $Measured\_Value_{ADC}$ is the actual Sensor Value converted by the PE5004 and n is the Number of filter steps. Values of 2, 4 or 8 for n are recommended as multiplication and division can be implemented with only little MCU time by shift operation.

4.3 EMI protection

The integrated lock in amplifier filters most of the external noise coupled in (see noise suppression). Nevertheless some peaks can be caused by devices radiating noise at nearly the same frequency, for example mobile phones. Beside the explained possibilities of avoiding noise by shielding or changing frequency, digital filter algorithms can be applied. As the phase overlap between noise source and lock in amplifier is only for a short period, EMI pulses occur as narrow spikes. A counting algorithm can be implemented additionally to the filter algorithm. Every time the threshold is reached, a counter is increased and continuous filter value is calculated (but without using it for further processing). If the threshold is under-run before the counter reaches a defined value, the counter is reset. If the counter end value is reached, the continuous filter value is passed for further processing and the counter is stopped until threshold under-run occurs.When being encountered with extreme EMI conditions, additional reliability can be achieved by spreading the used frequency spectrum and measuring in multiply steps, each at another frequency. This is fully supported by the PE5004.

4.4 Example Code

Example Code in C for a Measurement Routine including Filtering, negative Baselineupdate and EMI protection:

```
#define thresh 10               // touch threshold
#define thresh_neg 15           // threshold for negative baseline drift
#define neg _count 10           // min number of negative drift pulses for baseline update
#define pulsewait 5             // EMI Protektion - min number of pulses for touch recognition

// set all pointers to actual sensor start address, working with pointers helps saving mcu time when sampling sequent sensors , //
direct array addressing is realized by multiplication, this can consume a lot of time if no hardware multiplier is available

pCALIB= &SAR_CALIB[sens_start];          // pointer for calibration data (actual position baseline)
pTHRESH= &SAR_THRESH[sens_start];        // pointer for sensor threshold, only necessary if different threshold
                                         // values are used
pPULSE= &SAR_PULSE[sens_start];          // pointer for EMI pulse counter
pFILT= &SAR_FILT[sens_start];            // pointer for filtered sensor data
pBASE = &SAR_BASE[sens_start];           // pointer for continuous baseline calculation and sensor data calculation
pNEG= &SAR_NEG [sens_start];             // pointer for negative baseline drift counter

start_watchdog();
while (1){
      if (INT_PORT & INT){
            SPI_READ_5004_internal_sensors();            // SPI read command
            *pBASE= (((*pBASE)*3+SAR_DATA)>>2);          // continuous baseline and filter calculation, 4times filtering
            if (*pBASE-thresh_ neg >*pCALIB) (*pNEG)++;  // check for negative baseline drift (e.g. water drop),  increase
                                                         // baseline drift counter
            else *pNEG=0;
            if (*pNEG> neg _count){ *pCALIB=*pBASE; *pNEG=0;}      // check baseline drift counter and update baseline
            if (*pCALIB-*pTHRESH > SAR_DATA)  (*pPULSE)++;         // check sensor threshold, increase EMI counter
            if (*pPULSE>pulsewait){                                // check EMI counter
                        *pFILT=*pCALIB-*pBASE-*pTHRESH;            // pass filtered Value for further processing
                        *pPULSE=pulsewait+2;                       // hold counter on max
            }
            if (*pFILT==0) *pPULSE=1;                              // reset EMI counter on threshold under-run
            pCALIB++;                                              // increase all pointers
            pFILT++;
            pTHRESH++;
            pPULSE++;
            pBASE++;
            pNEG++;
             if (COL_ROW == COLEND_ROWEND)break;                   // break if last sensor is read
            while (INT_PORT & INT){                                // wait until interrupt low, only necessary for fast
                                                                   // communication >=1Mhz
                        SPI_READ_5004_internal_sensors();          // perform dummy read to assure last interrupt is low
                         if (TIFR0)break;                          // check for timeout
            }
      }
      if (TIFR0){break; terminal_puts("timeout\n");}               // check for timeout
```

Example Code in C for Baseline update and 42 Buttons:

```
if (base_step==0){
        timer16_on(baseupdate);
        base_step =1;
}
if (base_step==1) if (timer16_check()) {
        for (count=0; count<42;count++) SAR_BASE_save[count]=SAR_BASE[count];
        timer16_on(baseupdate);
        base_step=2;
}
if (base_step ==2) if (timer16_check()) {
        for (count=0; count<42;count++)
                if (SAR_BASE[count]<(SAR_BASE_save[count]+base_t))
                        if (SAR_BASE[count]>(SAR_BASE_save[count]-base_t))
                                SAR_CALIB[count]=SAR_BASE[count];
        base_step=0;
}
```

5 Combining multiple sensor arrays for large touch panels

Under normal circumstances a 10x10 sensor array is sufficient for a touch application, allowing an interpolated Resolution of 720x720 Points. For big touch displays it is possible to combine multiple arrays and PE5004. The following Layout allows a resolution of the touch position of 1440x1440 points over the whole matrix. The best way to communicate with each Chip is using I2C high Speed Mode. In addition one Interrupt Pin per PE5004 must be reserved. With the WAIT_ROW Register of the PE5004 additional time could be given in case the microcontroller is not fast enough to handle all of the sensor arrays during one measurement step (default is 100us per Row).

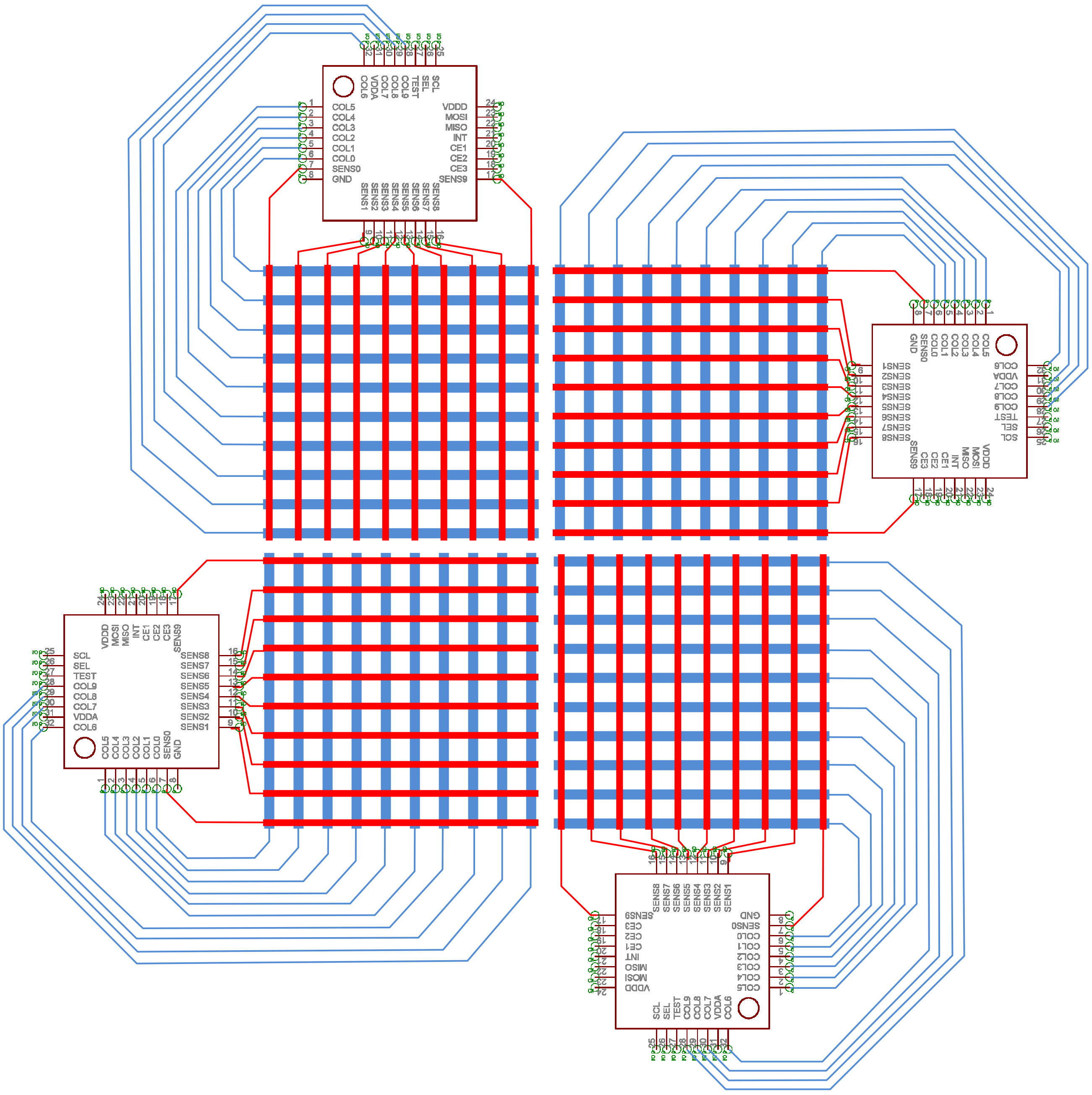

Figure 3 – 4x100 multiple matrix layout

6 Conclusion

The PE5004 integrated circuit provides functionality in conjunction with any simple 8 bit microcontroller that allows for robust capacitive sensing applications. It is unique in its abilities to account for:

- extreme EMI environments
- true capacitive multitouch screens
- high resolution up to 1440x1440 touch point applications
- any other capacitive matrix sensing application, like touch buttons or sliders
- ultra low power applications
- high precision sensing applications
- wet surface or other fast changing environmental condition applications

7 Contact

Germany

Stuttgart

Productivity Engineering
Process Integration GmbH
Behringstrasse 7
D-71083 Herrenberg
Germany
Phone.: +49 (0) 70322798 0
Fax: +49 (0) 70322798 29
Email: info@pe-gmbh.com
Web: www.pe-gmbh.com

Dresden

Productivity Engineering GmbH
Branch
Sachsenallee 9
D-01723 Kesselsdorf
Germany
Phone.: +49 (0) 35204777 00
Fax: +49 (0) 35204777 000
Email: info@pe-gmbh.com