SUPPORT NOTE

SN-123

Embedded User Function Control of Thermal Chuck

Introduction

If the features of Reedholm's integrated thermal chuck control are not required, a crude solution for setting and monitoring chuck temperature can be implemented at the user function level. This note provides guidance in using the low-level Temptronic thermal chuck driver inside an RDS Intranet embedded user function library file (DLL). It can be applied to other thermal chuck models using RDS Intranet, but there is no coverage of RDS DOS user functions.

Implementing the following features of Reedholm's integrated control is beyond the scope of a support note, and should only be attempted if the Reedholm software is not available.

- X & Y compensation at temperatures >80°C
- Automatic profiling at final temperature
- Lifting of probes during temperature ramp
- Probing wafer lots at multiple temperatures
- Computation of TCR using the TCR Wizard

Embedded User Function File

If one doesn't already exist, an embedded user function file must be created. If one exists, a new one can be created and used just for thermal chuck control as long as the processes that need such control do not need any other routines in the existing user function files. If multiple embedded user function files are used, thermal chuck control needs to be added to each one that needs it.

A new embedded user function file is created by copying USERNONE.PAS, an unpopulated example file. Since the DLL runs in MS-DOS, its filename is limited to eight characters and must match the library name entered at the top of the file.

Select Point DLL by Process

For the new embedded user function library (DLL) to be used during testing, Acquire must be configured to load it by modifying each desired RDS Intranet Process. The Point DLL filename is named the same as the embedded file.

Temptronic Thermal Chuck Driver

A source file for controlling the Temptronic thermal chuck is delivered with the RDS Intranet test engine. This driver file (T315FNC.PAS) is normally linked into a prober driver, but it can be linked into a user function file.

A copy of T315FNC.PAS must be moved from the test controller C:\RI\DRIVER directory to the embedded share directory C:\TCSHARE\EMBEDUSF\ on the test client or lab edition computer. Once moved, the file is linked into the user library using the Borland Pascal include ($I) compiler instruction.

The copied file also needs to be modified in order to execute properly inside a user function instead of a prober driver:

- UserFuncInc Boolean is set True

The two constants in figure 1 are needed by T315FNC, so they must be added to the user function file above the include statement. The Boolean variables AlreadySoaked and AlreadyInit are used by the user function code described in this note.

```
Const
  THCName = 'THC';
  PRB_DebugON = False;

Var
  ProberErr     : Integer;
  AlreadySoaked : Boolean;
  AlreadyInit   : Boolean;

(*$I T315FNC.PAS *)
```

Figure 1 – Top of User Function File

Creation of User Function Code

After including the T315FNC file and adding the required constants and global variables, the following modifications are made to the user function file:

1) Copy of routine GotoTemperature from USERFUNC.PAS added.
2) Figure 2 is the source for GotoTemperature after it is altered for the Temptronic thermal chuck instead of the prober's integrated HC.
3) Figure 3 is the source code for UserFunction_0, which is edited to call GotoTemperature.
4) Figure 3 also has the source code for UserLotStart, UserLotEnd, and UserWaferStart. Those are modified so that the thermal chuck is initialized per lot, permits soaking each wafer, and returns the stage to 30ºC at the end of the lot.

UserFunction_0 Parameters

To probe at temperature, the user function routine that calls GotoTemperature (UserFunction_0 in this example) must be called as the first test in the first intradie. Following are the test parameters used:

- CalcOne is target temperature in Celsius.
- CalcTwo is resolution in Celsius indicating that the temperature has been reached (e.g., 0.5).
- InitialDelay is per-wafer soak time in seconds.

RDS Intranet Version Supported

The user function code in this note is dependent upon the routine UserLotEnd called by Acquire. However, UserLotEnd is not called in RDS Intranet versions 1.10 and earlier. For version 1.11, patch RSPI0004 must be applied.

```
Function GotoTemperature(SetTemp, SetRes  : Double) : Double;

Var
  Temp                    : Double;
  Done, FirstSet, Ok, Quit : Boolean;

Begin
  Quit:=False;
  FirstSet:=False;
  Temp:=InvalidTest;
  Done:=False;
  Ok:=True;
  DisplayOneSlaveError('Setting temperature');
  Repeat
    Ok:=Get315Temp(THCname,Temp);
    DisplayOneSlaveError('Set='+ConvertNumber(SetTemp,'D') +
    '  Meas=' +ConvertNumber(Temp,'D'));
    If Ok=True Then
    Begin
      Done:=(Temp>=SetTemp-SetRes) And (Temp<=SetTemp+SetRes);
      If Done=False Then
      Begin
        If FirstSet=False Then
        Repeat
          Ok:=Set315Temp(THCname,SetTemp);
          If Ok=False Then Quit:= True;
          If (Quit=False) And (CalledFromAcquire=True)
           Then Quit:=GetV_QuitAcquire;
        Until (Quit=True) Or (Ok=True);
        FirstSet:=True;
        If Quit=False Then RI_Delay(1000);      (* Don't ask too often *)
      End;
    End Else
    Begin
      Done:= True;
    End;
    If (Quit=False) And (CalledFromAcquire=True)
     Then Quit:=GetV_QuitAcquire;
  Until (Done=True) Or (Quit=True);
  GotoTemperature:=Temp;
End; (* Function GotoTemperature *)
```

Figure 2 – GotoTemperature Source Code

```
(*_____________________________________________________
| FUNCTION  : UserFunction_0
| VERSION   : 1.00
| DATE      : March 9, 2005
|.....................................................
| DEFINITION/NOTES:
|   Inits the HC if not init this lot yet and then runs
|   to selected temperature. Only calls soak time once per
|   wafer.
|   CalcOne     : Set temp
|   CalcTwo     : Temp resolution (1C, 0.5C, etc)
|   InitialDelay: Soak time in seconds
|_____________________________________________________*)
Function UserFunction_0 : Double;

Begin
  MeasString:= 'UserFunction_0 result';
  ProberErr:=0;
  With Testplan^ Do                    (* Use TestPlan Parameters  *)
  Begin
    If AlreadyInit=False Then
     If Init315(THCname)=True Then
     Begin
       AlreadyInit:=True;
       UserFunction_0:=GotoTemperature(CalcOne,CalcTwo);
       If AlreadySoaked=False Then
       Begin
         RI_Delay(InitialDelay*1000);   (* Soak time is in seconds, not ms *)
         AlreadySoaked:=True;
       End;
     End Else
     Begin
       UserFunction_0:=2E+20;
     End;
  End;
End; (* Function UserFunction_0 *)

(*_____________________________________________________*)
Procedure UserLotStart; Export;

Begin
  PutV_QuitAcquire(Not LoadAquireProcedures);
  AlreadyInit:=False;
End; (* Procedure UserLotStart *)

(*_____________________________________________________*)
Procedure UserLotEnd; Export;

Var
  Tmp : Double;
Begin
  If AlreadyInit=True Then
  Begin
    Tmp:=GotoTemperature(30,1);  (* Return to 30 C, don't soak)
  End;
End; (* Procedure UserLotEnd *)

(*_____________________________________________________*)
Procedure UserWaferStart; Export;

Begin
  AlreadySoaked:=False;
End; (* Procedure UserWaferStart *)
```

Figure 3 – UserFunction_0, UserLotStart, UserLotEnd, and UserWaferStart Source Code