MotionCode™

Step-by-step solutions for motion applications

Galil MotionCode Solution:

Chip Encapsulation Material Fluid Dispensing Machine

Contents **Page**

1. Machine Description

Semiconductor manufacturing involves intricate automated operations from silicon crystal growth to final packaging. The packaging stage of these components is typically completed by the coupling and adhesion of two or more material layers around a silicon die. The application of the adhesive material between these layers is our point of interest.

This industry tool focuses on Chip Encapsulation Material dispensing, or "CEM". CEM fluid dispensing creates the bond layer between chip package layers after the die has been bonded to the package substrate. Fluid types include ceramic adhesives, conductive and non-conductive epoxies, thermal interface materials, and solder paste.

Fluid dispensing techniques require a minimum of two axes coupled tightly in a coordinate system, a third axis for vertical positioning of the fluid dispense nozzle, a fourth axis for conveyor motion, and a fifth axis for pump flow (*figure 1)*. Complex surface substrates often require coupling of the nozzle axis for three-dimensional architecture, and extremely complex surfaces may demand two additional axes coupled in a second coordinate system for yaw control of the dispensing nozzle. Achieving precise fluid dispensation commonly involves modes of motion such as linear interpolation and electronic gearing. In this application we will cover one coordinate system, four axes of motion control plus one additional for flow control. Although this is a specific example, the information contained within can be applied generally to fluid dispensing applications.

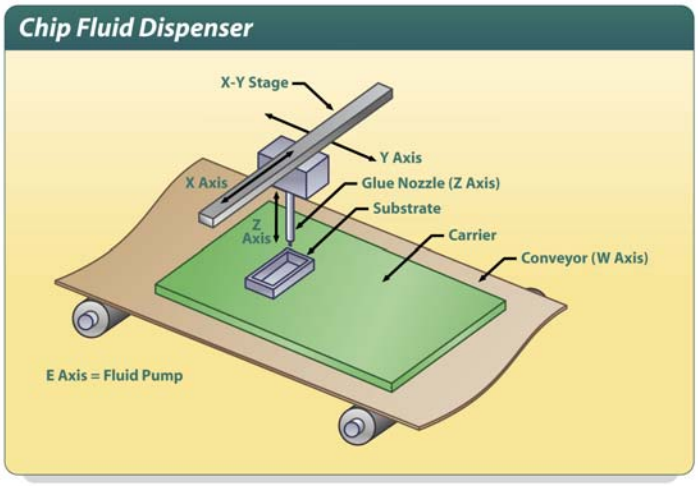

Figure 1.

2. Requirements

The minimum motion control requirements for the system are as follows:

(1) Five axes (XYZWE)
(2) Predetermined coordinated XY path
(3) Consistency of adhesive volume (precise flow control)
(4) I/O
New carrier sensor (digital input)
Carrier in position sensor (digital input)
Carrier unload sensor (digital input)
Fluid level monitoring (analog input)
Fluid level low alarm (digital output)
Forward/reverse limits for XYZ (digital inputs)
Home switch for W (digital input)
(5) Terminal User Interface
Operator enters substrate depth
Begin dispense cycle
Monitor machine cycle

3. Components Selected

This section describes the Galil hardware and software products chosen to implement the control system for the machine. Below is a complete bill of materials followed by a description of major components.

Table 1. Bill of Materials for Fluid Dispensing Control System

Part Number	Description	Unit Price (U.S.) Qty 1 / qty 100
DMC-1850	PCI Bus Motion Controller 5-axis	$2595 / $1345
CABLE-100-1m	High density cable in 1 meter length	$125 / $95 x2
AMP-19540	4-axis amplifier for driving four brush or brushless servos	$795 / $495
BLM-N23-50-1000 or equivalent	NEMA 23 Brushless Servo Motor, 1000 ppr encoder X4	Consult mfg.
CPS-12-56 or equivalent	Power Supply 12A, 56V	Consult mfg.
ICM-2900-FL	ICM-2900 with Flanges (Recommended for board level products)	$295 / $195
–OPTO Option	Specifies Opto-isolation to be added for General Outputs	$+50
CB-50-100-1880	100 Pin High Density / Ribbon Cable Adaptor.	$75 / $50
WSDK	Servo Tuning and Analysis Software	$195 (one time)

Controller: DMC-1850

Since the application requires a terminal operator interface, a dedicated PC is required. This makes the DMC-1850 PCI-Bus controller the optimal choice for controlling the 5 axes. The DMC-1850 also provides opto-isolated inputs for sensors and analog inputs for fluid level monitoring.

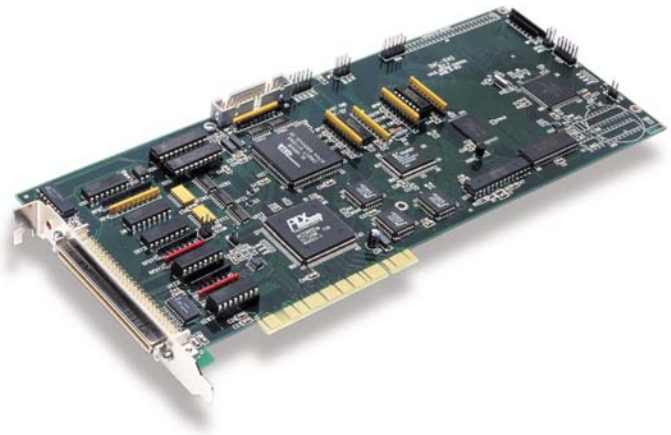

Figure 2. DMC-1850

Motor: BLM-N23-50-1000

For maintenance-free operation, we choose brushless motors. Galil's NEMA 23 #BLM-N23-50-1000 brushless motors, or equivalent, are appropriate because all axes require less than 0.3 Nm of continuous torque. Incremental encoders with 1000 cycles per revolution are installed on the motors resulting in 4000 quadrature counts per revolution. Hall sensors are not required on the motors as the incremental encoders provide commutation tracks for input to the amplifiers.

Amplifier: AMP-19540

To drive the X, Y, Z & W motors, we choose the very compact AMP-19540, which is a four-axis brushless amplifier (500 watts per axis). The fifth axis fluid pump requires proprietary drive equipment and so is instead connected to the controller via an ICM-2900 and CB-50-100, which breaks out axes 5-8 on a 5-8 axis controller.

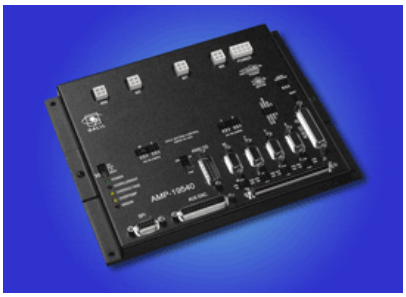

Figure 3. AMP-19540

4. Implementation

This section details how the components selected above were used to implement the control system.

Vector Mode

Vector mode coordinates two axes together for two-dimensional linear and arc segments. The linear and circular interpolation are computed by the controller, and vector speeds anywhere along the path can be specified to tailor the motion profile (see VP command < and > operators in ***command reference***). The code below generates the path shown in *Figure 4*.

Essential Commands: VM, VP, CR

```
VMXY                ;'Initiate Vector Mode
VP 4000,0           ;'specify first linear
                    segment of the substrate edge
CR 500,270,90       ;'specify first arc segment
                    of the substrate edge
VP 4500,2000
CR 500,0,90
VP 0,2500
CR 500,90,90
VP -500,500
CR 500,180,90
VE                  ;'end the sequence for the
                    substrate
BGS
```

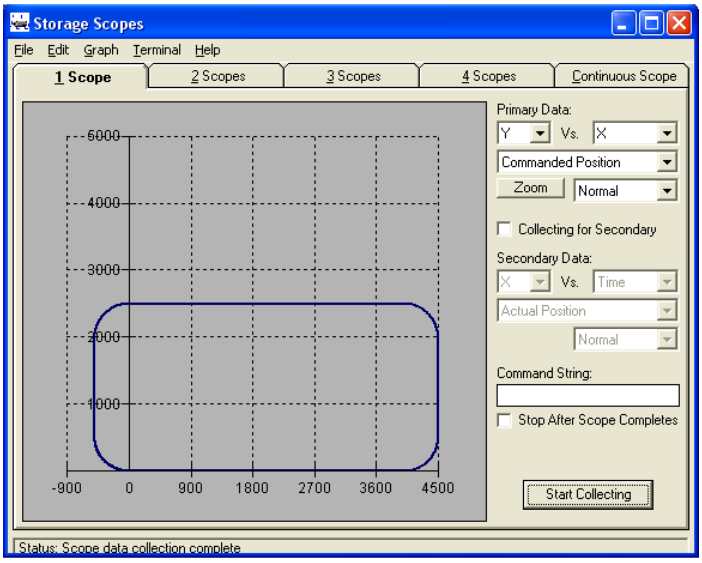

Figure 4. XY motion path during dispense captured with WSDK storage scope

Gearing

The flow control of the fluid dispensed from the nozzle is directly related to the speed of the XY vector path. To maintain a consistent dispense volume per unit of travel, we gear the pump axis to the arc length along the XY path. The controller will automatically adjust the dispense flow without us having to consider the effects of acceleration and deceleration along the path in the application code.

Essential Commands: GA, GR

```
GAE=S               ;'Specify the S coordinate
                    plane as the master for pump
                    axis, E

GRE=0.135           ;'Specify the ratio of pump
                    flow to vector speed
```

Fluid Level Monitoring via Threads

The machine needs to monitor the fluid level to maintain proper control of the process. A loop running in a second thread continuously scans an analog input to monitor the fluid level. If the fluid level is too low, it stops the machine and alerts the operator by triggering an alarm.

```
XQ #PumpLvl, 1      ;'begin monitoring pump fluid
                    level in a second thread

#PumpLvl            ;'this code monitors the pump
                    fluid level via analog input.
  IF @AN[1] < 1
    MG "FLUID LEVEL LOW"
    SB1             ;'output 1 sets an alarm for
                    low fluid level

    AB
  ENDIF
JP #PumpLvl
```

Terminal Operator Interface (IN and MG commands)

The machine uses a very simple text terminal interface. The MG command is used to print status messages to the screen and the IN command is used to collect user input (in this case the distance the Z axis needs to move to reach the substrate). *Figure 5* shows a terminal session with the machine.

```
:Homing all axes...
Place new carrier on conveyor
Specify substrate depth (counts)
1000
Moving carrier to start point...
Dispensing...
Dispense has ended
Moving carrier to end point...
Remove finished carrier
Place new carrier on conveyor
```

Figure 5. Terminal session with the machine

Program Code

This section shows the Galil-language program used to run the machine. The program is burned into the controller with the BP command. When the PC and controller are powered up, the application code immediately begins from the special program label #AUTO. It first starts up the fluid level monitoring thread #PumpLvl (that runs in parallel with #AUTO) and then homes the XY stage, the nozzle, and the conveyer. After performing this one-time initialization, #AUTO goes into the infinite #Loop, which performs the main dispensing operation.

It starts by prompting the operator to insert a new carriage. After the new carriage sensor (input 1) detects the carriage is in place, the program prompts the user for the substrate depth. After this data is entered, the carriage is moved until the in-position sensor trips (input 2), at which time the dispensing pattern executes *(figure 4)*. When the pattern is done, the carriage is moved until it hits the end of travel sensor (input 3), and last the program waits until the user removes the carriage before starting again.

```
#AUTO
  PmpRatio = 0.1     ;'pump gear ratio for fluid flow based off
                      'of s plane velocity as the master axis

   CB1               ;'clear fluid level alarm
   XQ #PumpLvl, 1    ;'begin monitoring pump fluid level in a second thread

   'home the nozzle axes XYZ to the reverse limit switch and then the index pulse
   'home the conveyor W to the home switch and then the index pulse
   MG "Homing all axes..."

   JG*=-10000        ;'jog towards reverse limit switch
   HMW               ;'home conveyer to home switch (CN specifies direction)
   BGXYZW            ;'begin motion towards limit (conveyer to home)
   AMXYZW            ;'wait until we hit the limit (index for the conveyer)

   JGXYZ = 500       ;'move slowly towards the index pulse
   FIXYZ             ;'find index
   BGXYZ             ;'begin motion towards index
   AMXYZ             ;'wait until we hit the index.Position is set to 0.

   #Loop
      MG "Place new carrier on conveyor"
      AI-1           ;'wait for carrier positioning. carrier placed at start area of
                      'conveyor activates sensor at input 1

      'asks operator to enter nozzle height spec for new substrate.

      IN "Specify substrate depth (counts)",ZPos

      'position conveyor to line up carrier with sensor on input 2 indicating
      'start position
      MG "Moving carrier to start point..."
      JGW = 500
      BGW
      AI-2
      STW
      AMW

      JS #Dispens
      'send carrier to finished end of conveyor, and wait until it hits
      'the end of carrier switch (input 3) before stopping
      MG "Moving carrier to end point..."
      JGW = 5000
      BGW
```

(Continued next page)

(Continued from page 4)

```
      AI-3
      STW

      MG "Remove finished carrier"
      AI3                ;'wait for carrier-end-of-travel input 3 to go back high from removal of
                          'finished carrier.
   JP #Loop              ;'jumps back to a previous program section for conveyor
                          'run to next carrier.

EN

#Dispens
   'position the nozzle to the specified distance from substrate surface
   PAZ = ZPos
   BGZ
   AMZ

   GAE = S               ;'gear the pump axis to the vector sequence
   GRE = PmpRatio        ;'the ratio of pump feed to the vector speed as specified

   VMXY                  ;'initiate vector mode
   VA 250000             ;'set acceleration, deceleration, and speed parameters
   VD 250000
   VS 5000
   VT 0.8                ;'and add smoothing constant

   'substrate path defined here
   VP 4000,0             ;'specify first linear segment of the substrate edge
   CR 500,270,90         ;'specify first arc segment of the substrate edge
   VP 4500,2000
   CR 500,0,90
   VP 0,2500
   CR 500,90,90
   VP -500,500
   CR 500,180,90
   VE ;'end the sequence for the substrate

   MG "Dispensing..."
   BGS
   AMS

   PAZ = 0               ;'move nozzle up off substrate after path has stopped
   BGZ
   AMZ

   MG "Dispense has ended"
EN

'*************************MONITORING ROUTINE*****************************

#PumpLvl                 ;'this code monitors the pump fluid level via analog input
   IF @AN[1] < 1
      MG "FLUID LEVEL LOW"
      SB1                ;'output 1 sets an alarm for low fluid level
      AB
   ENDIF
JP #PumpLvl
EN
```