BAB III

PERANCANGAN SOFTWARE

3.1. Langkah Perancangan

Pada perancangan *software* simulasi CNC ini dilakukan dengan mengklasifikasikan semua proses–proses mesin CNC ke dalam unit–unit tertentu, dalam tiap unitnya terdiri dari modul–modul perintah. Cara ini lebih memudahkan dalam pembuatan *software*, karena pada *form* utama tidak terlalu banyak memuat perintah pemrograman. Pada perancangan sistem ini terdapat dua *form* utama, yaitu *Form User Interface* (*keyboard panel*) dan *Form* Animasi, dan tiap *form* terdiri dari unit modul. Setiap unit modul akan digabungkan antara satu dengan lainnya, sehingga *form panel* yang terdiri dari unit data dapat diakses oleh *form* animasi.

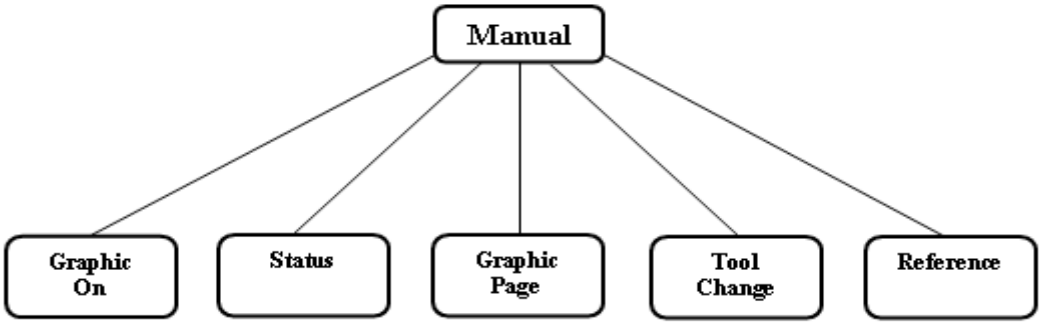
Prinsip kerja dari program *simulator* ini adalah menterjemahkan *file* program dari hasil *input user interface* (*keyboard panel*), dalam hal ini meliputi *file* pemrograman perintah G *codes* dan parameter–parameternya, M *codes*, *file tool* (TO). Selanjutnya setiap kode instruksi hasil *input* dieksekusi oleh unit animasi sesuai dengan fungsi pergerakannya.

3.1.1. Perancangan *User Interface* (*keyboard panel*)

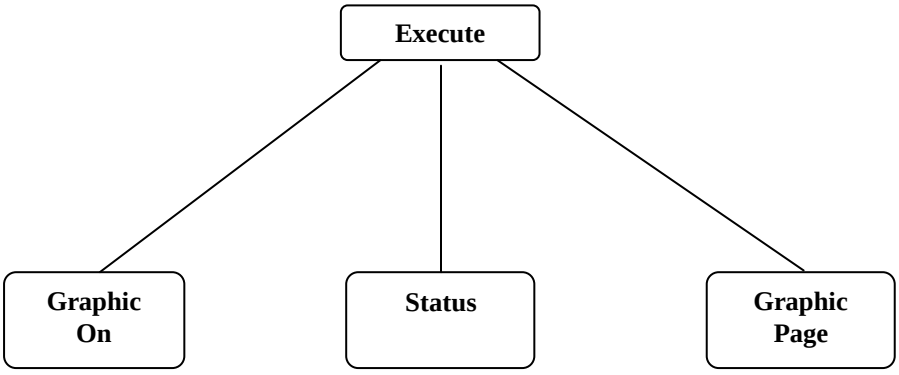
Pada bagian ini sistem mengklasifikasikan prosedural dari setiap tombol (panel) yang terdapat pada *keyboard* panel mesin secara keseluruhan. *Keyboard* panel simulasi terdiri dari empat mode utama, yaitu : *mode automatic*, *mode edit*, *mode execute*, *mode manual*. Setiap *mode* tersebut diklasifisikan sebagai subrutin pada *software* simulasi ini yang merupakan prosedur untuk memberikan perintah pemrograman.

User interface berisi modul–modul yang dapat menterjemahkan setiap perintah dari hasil *input* panel. Untuk memudahkan pembuatan program ini perlu adanya inisialisasi tiap modul dalam panel pada *user interface*, sehingga setiap hasil *input* dapat diterjemahkan menjadi perintah pemrograman pada setiap mode utama maupun *sub–mode*. Dengan cara di atas maka prosedur pemrograman dapat berjalan dengan terstruktur.

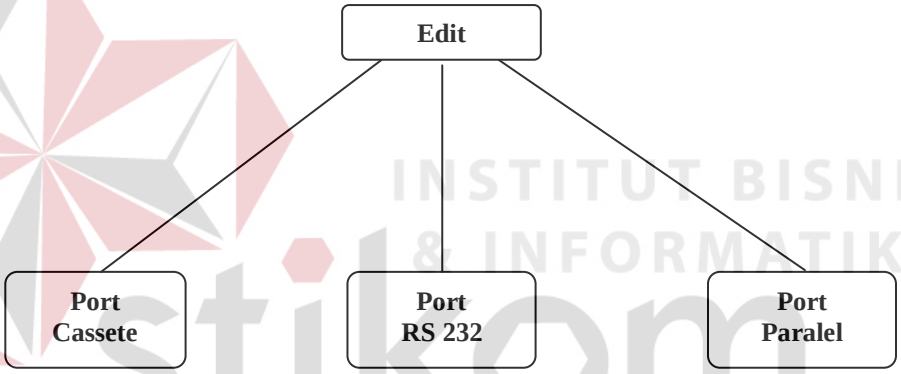
Adapun mode utama dan sub–mode (*softkey*) pada *keyboard* dapat ditunjukkan pada bagan dibawah ini :



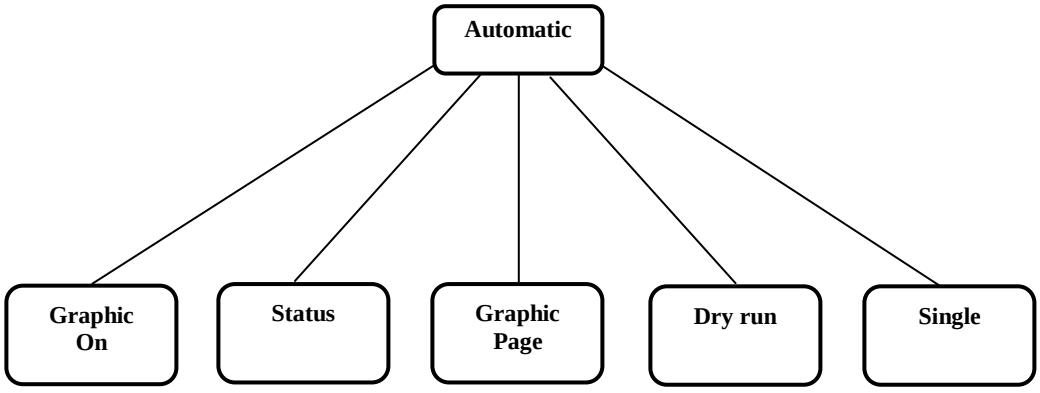
Gambar 3.1. Panel mode manual



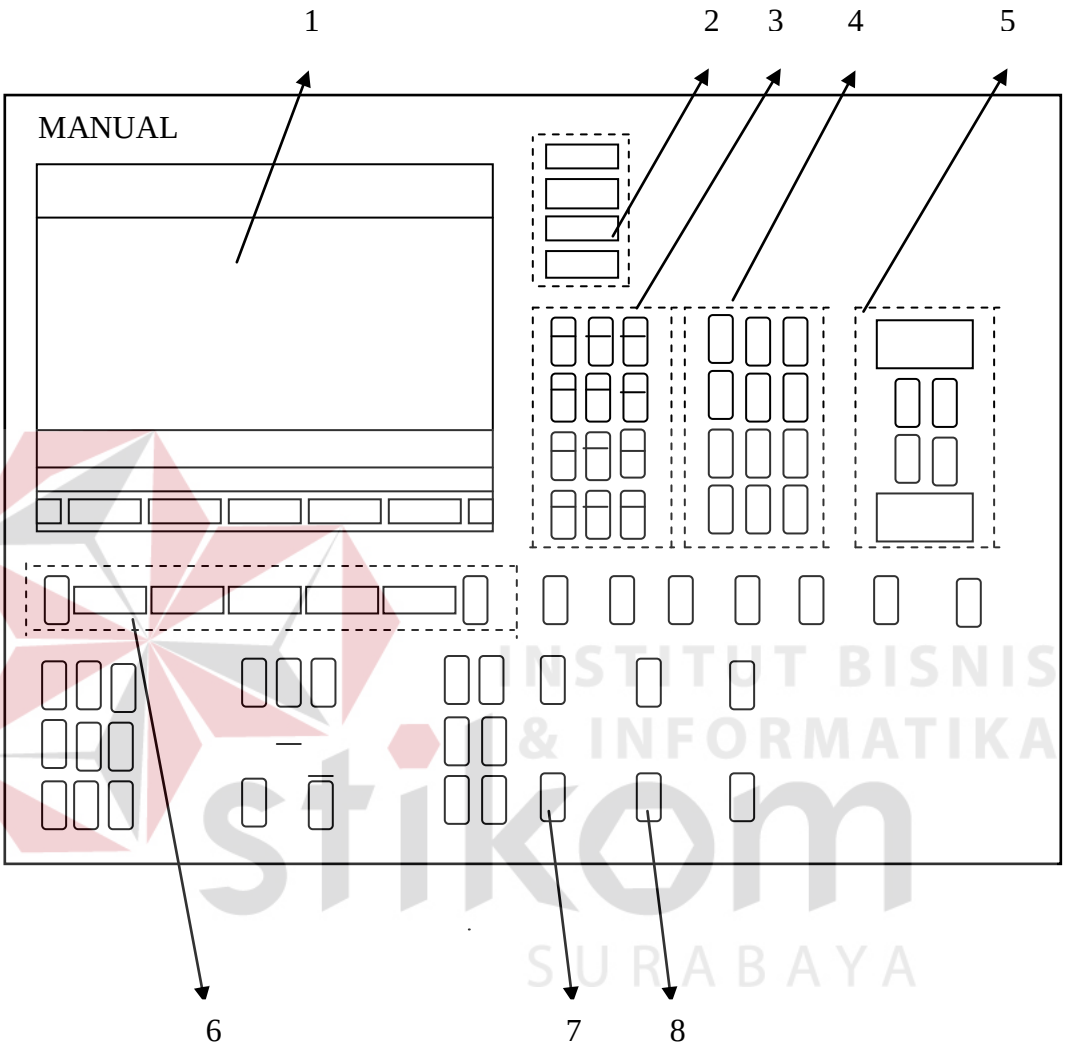
Gambar 3.2. Panel mode *execute*



Gambar 3.3. Panel mode *edit*



Gambar 3.4. Panel mode *automatic*



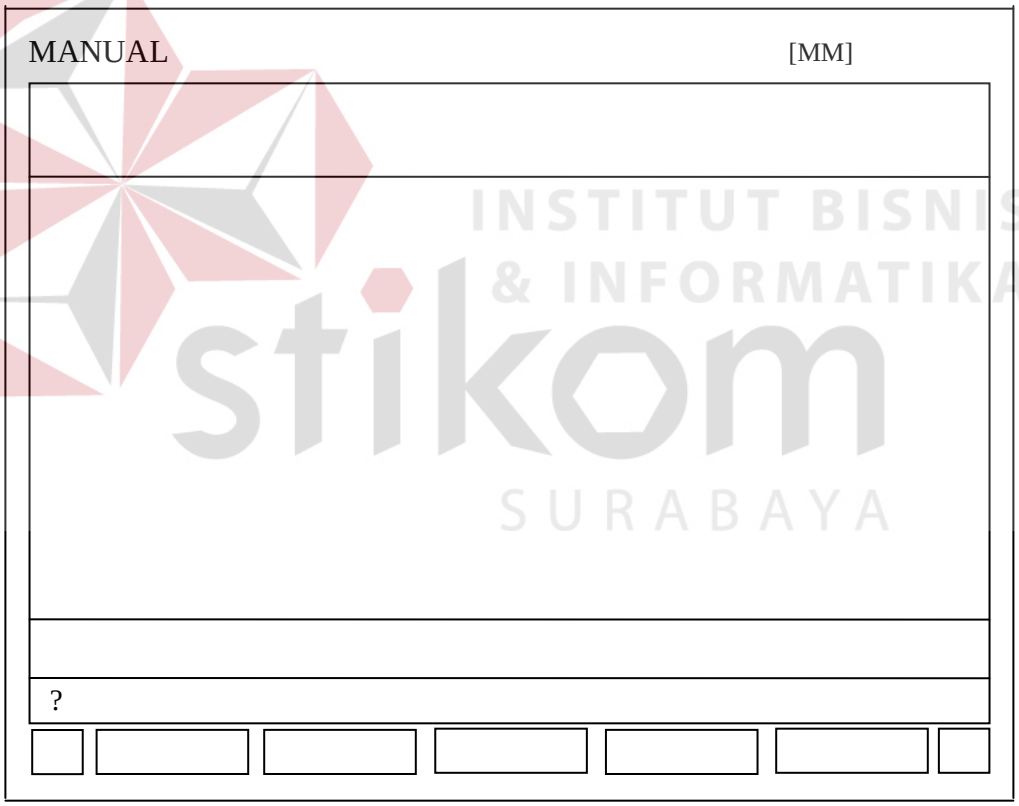
Gambar 3.5. Tampilan *keyboard* mesin CNC

1. Monitor
2. Mode *keyboard*
3. *Address keyboard*
4. Digit *keyboard*
5. *Keyboard function*
6. *Softkeys*
7. *RESET key*
8. *CYCLE START key*

Keyboard panel mesin (*user interface*) dalam bahasan *software* ini ditunjukkan pada gambar di atas. Pada *keyboard* terdapat bagian dan pengelompokan tombol menurut fungsinya, yang ditunjukkan dengan nomor di atas.

A. Monitor

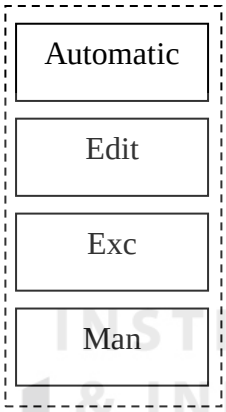
Bagian ini menampilkan informasi dan hasil *input* dari *user interface software* simulasi ini sesuai dengan mode utama yang aktif.



Gambar 3.6. Monitor

B. Mode *Keyboard*.

Bagian ini berfungsi sebagai pengaktifan mode utama mesin. Mode utama pada operasional mesin. Seperti yang sudah dijelaskan pada bagan di atas, ada empat mode utama pada panel, setiap mode terdiri dari beberapa sub mode. Kelompok dari panel mode utama dapat ditunjukkan pada gambar di bawah ini :



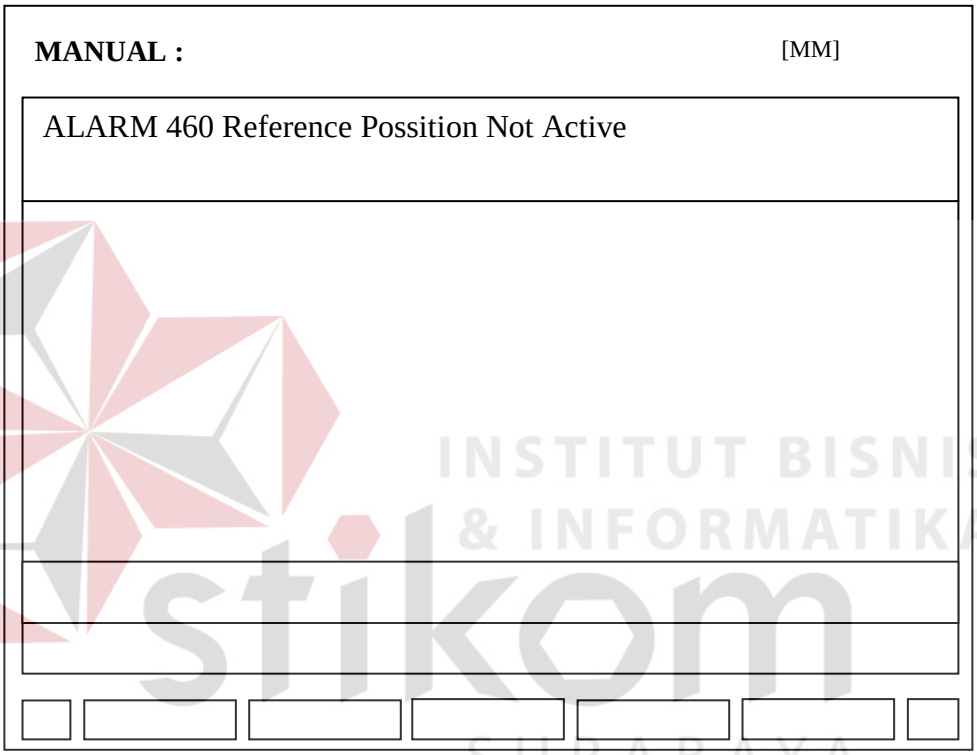
Gambar 3.7. Tombol mode

B.1. Mode Manual

Mode ini berfungsi sebagai mode pelayanan manual, dengan kata lain digunakan untuk melakukan proses pemesinan secara manual. Misalnya, pada mode ini dapat menjalankan pahat dengan cara manual dengan fungsi tombol *Manual Jog* bersamaan dengan arah panah .

Pada tahap perancangan *software* simulasi ini, tidak semua sub mode pada mode manual diaktifkan. Adapun yang dibahas pada *software* ini adalah pengaktifan titik referensi pada sub mode referensi. Sub-mode referensi yang dimaksudkan adalah untuk pencapaian titik referensi.

Pada pencapaian titik referensi ini sangat penting dalam hubungannya dengan data pergeseran titik nol mesin (*Position Shift Offset*) pada bahasan mode Utama *Edit*. Tampilan monitor pada mode manual ini dapat dilihat seperti gambar dibawah ini :



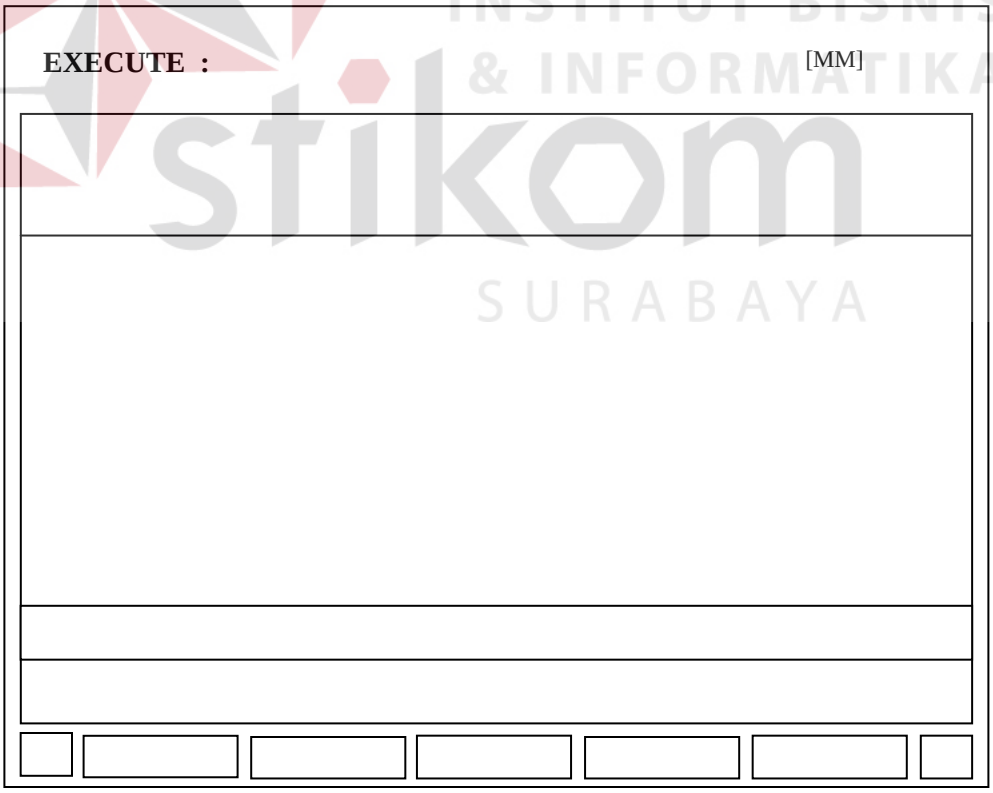
Gambar 3.8. Mode manual

Listing program tampilan manual :

```
void __fastcall TMainForm::bManClick(TObject *Sender)
{
  ScanKey(kMan, kMode);
  ShowAlarm();
  UpdateTitle("");
  UpdateTool();
}
```

B.2. Mode *Execute*

Mode ini berfungsi untuk pemrosesan *buffer* penyimpan blok, pemanggilan alat potong dan penggeserannya, sehingga harganya tersajikan pada mode manual, penggeseran pahat secara manual dengan harga *incremental* dengan panel *Manual Jog*. Pada pembahasan *software* ini, *mode Execute* tidak diaktifkan karena hanya digunakan fungsinya secara manual oleh *operator* dalam hal *setting* pahat. Seperti penjelasan diatas maka pada mode ini kami tetap memberikan tampilan di monitor *user interface*, akan tetapi fungsi–fungsi dari mode utama dan sub modenya tidak di bahas atau diaktifkan. Tampilan pada layar monitor dapat dilihat pada gambar berikut ini :



Gambar 3.9. Tampilan mode *execute*

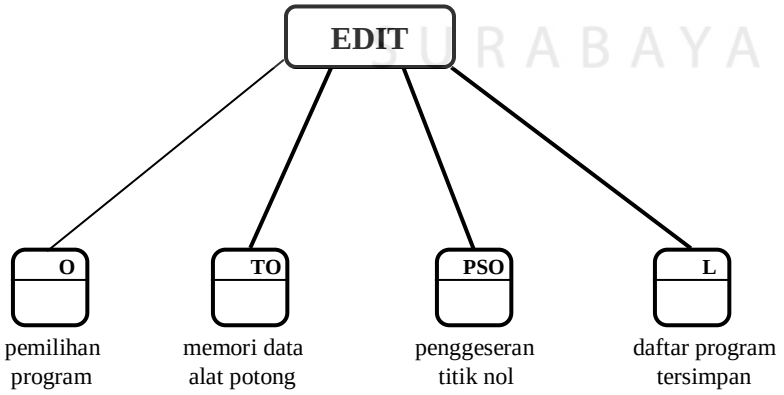
Listing program tampilan mode *execute* :

```
void __fastcall TMainForm::bExcClick(TObject *Sender)
{
  ScanKey(kExc, kMode);
  //  ResetDisplay();
  ShowExc();
  UpdateTitle("");
  UpdateTool();
}
```

B.3. Mode *Edit*

Pada mode edit ini ada beberapa kemungkinan antara lain :

1. *Input* program CNC (G codes beserta parameter–parameternya) lewat *keyboard* simulasi.
2. Memasukkan data alat potong.
3. Pengubahan data pergeseran titik nol (PSO).
4. Pemilihan program dan melihat daftar dari program yang tersimpan pada memori utama. Seperti yang dapat dilihat pada bagan di bawah ini :



Gambar 3.10. Mode *edit*

TO, PSO, L adalah aplikasi sub-rutin yang terdapat pada mode utama *edit*. Sedangkan untuk O (pemilihan atau pengukuhan nomor program) dapat diaplikasikan langsung pada saat kita sudah siap untuk memasukkan atau mengubah program CNC pada mode utama *edit*.

Pembahasan *software* yang dirancang ini perlu diperhatikan untuk sub-mode pada Mode Edit seperti, *port kaset, port RS 232,* dan *port paralel* tidak diaktifkan, fungsi dari sub-mode tersebut adalah untuk pemuatan dan pembacaan data melalui kaset, peralatan *RS 232,* dan antar peralatan paralel. Pada gambar 3.11 dapat dilihat dimana pada saat mode utama *edit* aktif monitor menampilkan tampilan tersebut. Pada saat itu juga kita dapat memasukkan program CNC ini dengan menekan panel pada *user interface* (*keyboard panel*).

Gambar 3.11. Tampilan mode *edit*

Listing program tampilan mode *edit* :

```
void __fastcall TMainForm::bEditClick(TObject *Sender)
{
  ScanKey(kEdit, kMode);
  //ResetDisplay();
  ShowEdit();
  UpdateTitle("");
  UpdateTool();

}
```

Sedangkan untuk fungsi subrutin pada mode utama edit untuk mengubah dan memasukkan data alat potong pada memori utama sesuai dengan prosedur yang berlaku dimesin dapat dilihat pada gambar 3.12.

EDIT : [MM]

	TO			
No	X	Z	R	L
00	0.000	0.000	0.000	0
01	0.000	0.000	0.000	0
02	0.000	0.000	0.000	0
03	0.000	0.000	0.000	0
04	0.000	0.000	0.000	0
05	0.000	0.000	0.000	0
06	0.000	0.000	0.000	0
07	0.000	0.000	0.000	0
08	0.000	0.000	0.000	0
09	0.000	0.000	0.000	0

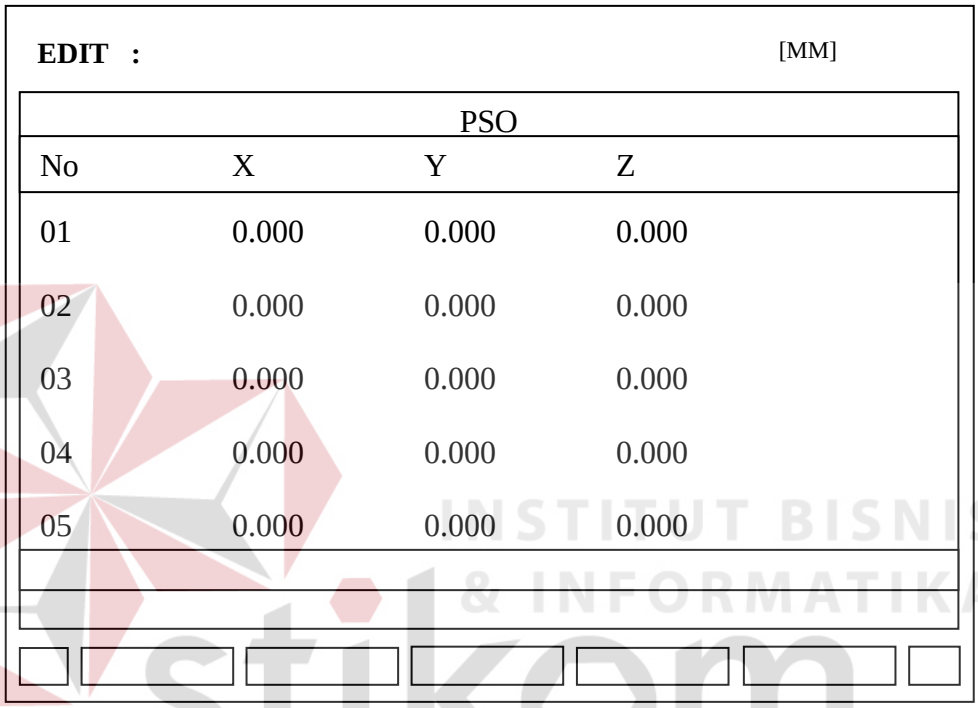
Gambar 3.12. Data alat potong (TO)

Listing program TO :

```
void __fastcall TMainForm::ShowTo(void)

{
 TempDataTo->Lines->LoadFromFile("TO.TXT");
 TextCanvas->Font->Color=clRed;
 TextCanvas->Brush->Color=clBlack;
 TextCanvas->Brush->Style=bsSolid;
 TextCanvas->Pen->Color=clBlack;
 TextCanvas->Font->Size=10;
 TextCanvas->Rectangle(10,78,10+467,78+200);
 /*
   TextCanvas->Brush->Color=clGray;
   TextCanvas->Rectangle(11,(80+15*(RowActive-dif)),11+3+TextCanvas->TextWidth(TempData->Lines-
>Strings[RowActive]),(80+15*(RowActive-dif))+TextCanvas->TextHeight("N"));
   TextCanvas->Brush->Color=clBlack;
   TextCanvas->Brush->Style=bsClear;
 */
 AnsiString label[5]={
                      "No:",
                      "X",
                      "Z",
                      "R",
                      "L"
                     };
  for(int j=0;j<=4;j++)
   TextCanvas->TextOutA(11+100*j,80, label[j]);
   TextCanvas->Pen->Color=clRed;
   TextCanvas->MoveTo(10,110);// 10,78,10+467,78+200
   TextCanvas->LineTo(477,110);
 for(int i=0;i<10;i++)
 {
    GetBuffOf(TempDataTo->Lines->Strings[i]);
    for(int j=0; j<=4; j++)
    {
      TextCanvas->TextOutA(11+100*j,120+15*i, TO.databuff[j]);
    }
 }

 WorkCanvas->CopyRect(Rect_Text,TextCanvas,Rect_Text);
 PaintBox1->Canvas->CopyRect(Rect_Back,WorkCanvas,Rect_Back);
}
```

Pada mode *edit* utama ini, fungsi untuk memasukkan dan mengubah data *position shift offside* (PSO) atau disebut pergeseran posisi titik nol, seperti yang terlihat pada gambar 3.13. Pencapaian titik referensi ini dimaksudkan sebagai referensi titik nol pahat pada saat pengerjaan benda kerja sehingga parameter yang dimasukkan selalu dihitung dari titik referensi tersebut (*absolute*).

Pada mode utama ini disediakan *file memory* untuk menyimpan data dari program G *codes*, PSO dan TO. Sehingga setiap aplikasi diaktifkan data terakhir yang tersimpan bisa dipanggil kembali.



Gambar 3.13. *Edit* PSO

Listing program PSO :

```
void __fastcall TMainForm::ShowPso(void)
{
 TempDataPso->Lines->LoadFromFile("PSO.TXT");
 TextCanvas->Font->Color=clRed;
 TextCanvas->Brush->Color=clBlack;
 TextCanvas->Brush->Style=bsSolid;
 TextCanvas->Pen->Color=clBlack;
 TextCanvas->Font->Size=16;
 TextCanvas->Rectangle(10,78,10+467,78+200);
 AnsiString label[4]={
                      "No:",
                      "X",
                      "Y",
                      "Z",
                     };
  for(int j=0;j<=3;j++)
   TextCanvas->TextOutA(11+100*j,80, label[j]);
```

```
  TextCanvas->Pen->Color=clRed;
  TextCanvas->MoveTo(10,110);// 10,78,10+467,78+200
  TextCanvas->LineTo(477,110);
 for(int i=0;i<5;i++)
 {
    GetBuffOf(TempDataPso->Lines->Strings[i]);
    for(int j=0; j<=3; j++)
    {
      TextCanvas->TextOutA(11+100*j,120+30*i, PSO.databuff[j]);
    }
 }

 WorkCanvas->CopyRect(Rect_Text,TextCanvas,Rect_Text);
 PaintBox1->Canvas->CopyRect(Rect_Back,WorkCanvas,Rect_Back);
}
```

B.4. Mode *Automatic*

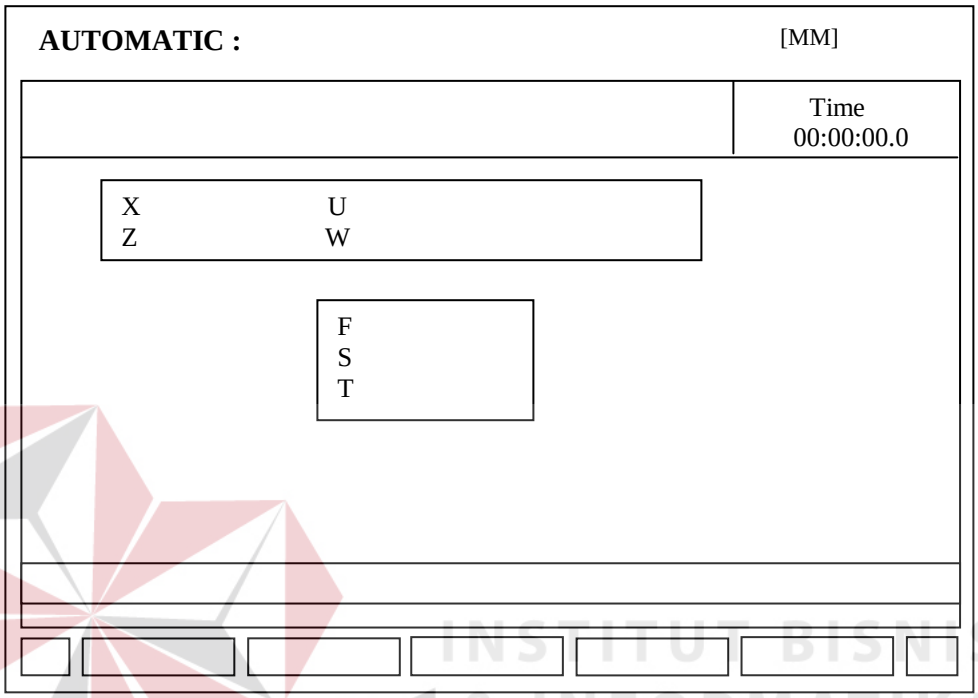
Pembahasan terakhir dari mode utama pada *user interface software* ini adalah mode *automatic*. Mode ini berfungsi untuk menjalankan program yang sudah disusun pada mode edit dan tersimpan pada *buffer* memori utama dari *software*.

Setelah program G codes siap untuk disimulasikan (tersimpan pada *buffer* utama), maka mode utama dari *automatic* harus diaktifkan terlebih dahulu. Setelah itu dimasukkan nilai dimensi dari benda kerja. Langkah selanjutnya mengaktifkan *form* animasi, kemudian untuk menampilan simulasi gerakan pahat dengan menjalankan perintah *cycle start*. Sajian informasi pada mode utama ini meliputi harga dari parameter yang dimasukkan pada tiap blok program, harga asutan pada benda kerja (F), harga kecepatan sumbu utama (S), nomor pahat dan nomor koreksinya.

Listing program tampilan mode *automatic :*

```
void __fastcall TMainForm::bAutomaticClick(TObject *Sender)
{
 ScanKey(kAuto, kMode);
 ShowAuto();
 UpdateTitle("");
 UpdateTool();
}
```

Tampilan layar monitor di mode *automatic* utama, seperti gambar di bawah ini :



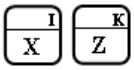
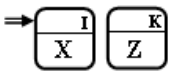
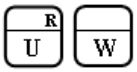
Gambar 3.14. Mode *automatic*

Program simulasi ini dirancang rancang pada saat *form* animasi aktif (simulasi proses pemesinan dijalankan), akan muncul *window* baru yang berisi *form* animasi tersebut. Secara bersamaan dengan itu dapat juga ditampilkan *form user interface* pada aplikasi *window* yang lain. kita dapat melihat blok program yang sedang aktif menjalankan proses selain di *window form* animasi dan sajian informasi–informasi yang lain seperti yang sudah kami jelaskan diatas.

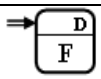
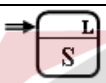
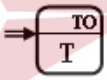
C. Address Keyboard

Address keyboard berfungsi untuk memberikan *input* perintah berupa alamat untuk memanggil aplikasi sesuai prosedur mesin yang sudah diklasifikasikan pada unit dalam *form* utama panel dalam *software* simulasi ini. Unit tersebut berisi modul–modul, dibawah ini beberapa jenis dari alamat–alamat pada *user interface* (panel) yang kami bahas pada perancangan *software* simulasi ini,

Tabel 3.1.a. Tabel tombol a*ddress* beserta fungsinya

Tombol Address	**Hasil Tombol**	**Keterangan**
	N	Alamat untuk nomor blok
	O	Untuk nomor program
	G	Untuk fungsi G
	PSO	Pergeseran posisi titik nol atau memasukkan ke tabel PSO
	M	Untuk fungsi M
	P	Untuk parameter dalam siklus
	X, Z	Untuk data gerak dalam absolut
	I, K	Untuk parameter titik pusat lingkaran
	U, W	Untuk data gerak dalam inkremental

Tabel 3.1.b. Tabel tombol *address* beserta fungsinya

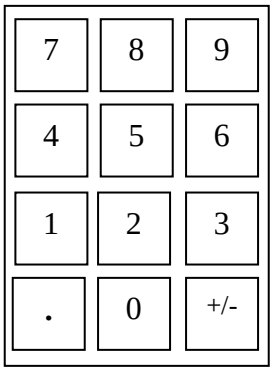
Tombol Address	Hasil Tombol	Keterangan
	F	Untuk kecepatan asutan pada benda kerja
	D	Untuk parameter dalam siklus
	S	Untuk kecepatan putaran sumbu utama
	L	Untuk parameter dalam siklus
	T	Untuk alamat alat potong
	TO	Untuk alamat masuk pada data pahat

Keterangan : tanda panah [⇒] menunjukkan bahwa *keyboard* fungsi SHIFT aktif

D. Digit Keyboard

Digit keyboard berfungsi untuk memasukkan data nilai dari setiap alamat yang bersifat perintah pada pemrograman *G codes* beserta parameter, misal nilai dari koordinat X, Z, U, W dan harga dari F, S, T, dan parameter lain. Pada *software* ini nilai desimalnya adalah tiga digit dibelakang parameter koma. Dan satuan yang digunakan adalah milimeter.

Digit *keyboard* pada *user interface* (panel mesin) dalam perancangan *software* ini dapat dilihat pada gambar13.15.



Gambar 3.15. *Digit keyboard*

E *Keyboard Function*

Keyboard function adalah fungsi pengendali. Pada pembahasan perancangan ini dapat dijelaskan berbagai *keyboard* fungsi beserta penggunaannya pada saat mengoperasikan *software* simulasi ini. Adapun penjelasannya sebagai berikut,

1. Tombol *Enter*, digunakan untuk :
   a) Menyimpan ke memori utama
   b) Menetapkan nomor program baru
   c) Memindahkan kursor yang aktif
   d) Pemanggilan pencatat TO, PSO, N, O
2. Tombol *Shift*, berfungsi sebagai pengalihan fungsi tombol yang aktif.
3. Tombol *Store Next*, berfungsi menyimpan blok program ke dalam *file* memori utama.
4. Tombol *Previous*, berfungsi untuk memindahkan kursor ke blok yang aktif.
5. Tombol *Clear entry*, berfungsi untuk menghapus data terakhir (angka).

6. Tombol *Clear block*, berfungsi untuk menghapus blok dalam *file* memori.
7. Tombol *Clear word*, berfungsi untuk menghapus satu kata (kursor yang aktif harus pada posisi kata yang akan dihapus).
8. Tombol *Clear program*, berfungsi untuk menghapus program yang tersimpan pada *file* memori utama.

F. *Reset Key* dan *Cycle Start Key*

Panel *reset key* berfungsi sebagai *reset* atau kembali pada mode utama, panel ini digunakan apabila *user* ingin membatalkan proses yang terlanjur dilakukan. Sedangkan panel *cycle start key* berfungsi untuk memulai proses animasi dan pengaktifan referensi titik nol mesin setelah penekanan panel *reference*.

3.1.2. Perancangan Subrutin PSO

Pada perancangan subrutin PSO ini digunakan *buffer* penyimpan memori utama. Pada subrutin ini *buffer* dibagi menjadi dua antara lain : *buffer* PSO (a), yang berfungsi sebagi penyimpan data pergeseran titik nol untuk fungsi G54 dan G55, sedangkan untuk *buffer* PSO (b) untuk menyimpan data pergeseran G57, G58 dan G59. Pembagian *buffe*r PSO menjadi dua untuk membantu peterjemahan posisi akhir sebelum proses fungsi pembatalan (G53 dan G56) yang aktif dan melakukan pembatalan pergeseran titik referensi nol.

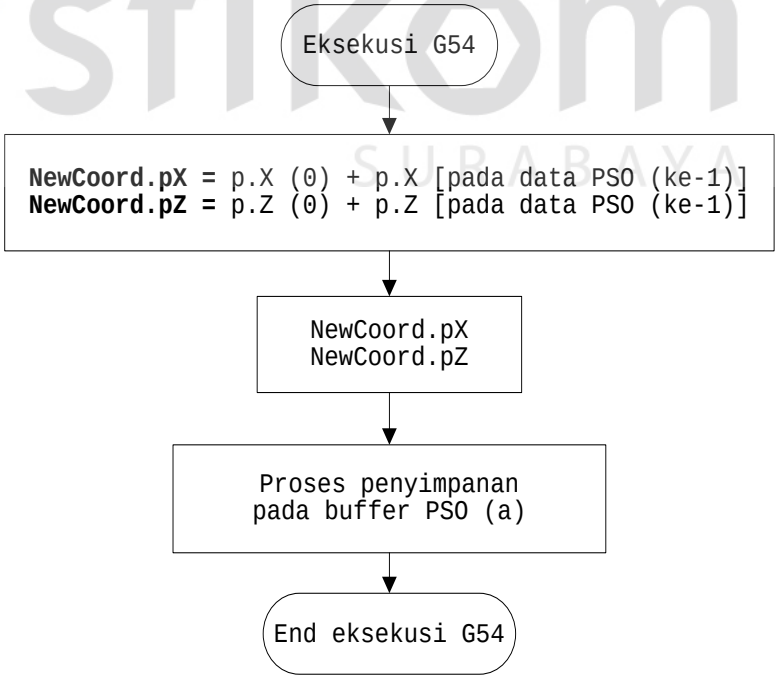
Pada subrutin PSO ini terdapat beberapa fungsi G yang mempunyai peran dalam perintah pergeseran posisi titik nol referensi. Perintah pergeseran posisi titik nol ini dicatat dalam baris-baris yang menunjukkan harga nilai koordinat X dan Z, dari perintah pergeseran kode G tersebut, seperti yang ditunjukkan gambar 3.16.

	PSO			
	No	X	Y	Z
G54	1	0.000	0.000	50.000
G55	2	0.000	0.000	70.000
G57	3	0.000	0.000	0.000
G58	4	0.000	0.000	0.000
G59	5	0.000	0.000	0.000

Gambar 3.16. Pencatat PSO

Dari gambar di atas dapat dilihat bahwa tiap-tiap fungsi G memiliki baris sendiri dalam pencatatannya di subrutin PSO. Untuk mengetahui proses pemanggilan dan eksekusi perintah G dalam PSO pada *software* ini dapat dilihat dari beberapa diagram alir berikut :

A. Eksekusi Kode G54



Gambar 3.17. *Flow chart* eksekusi kode G54

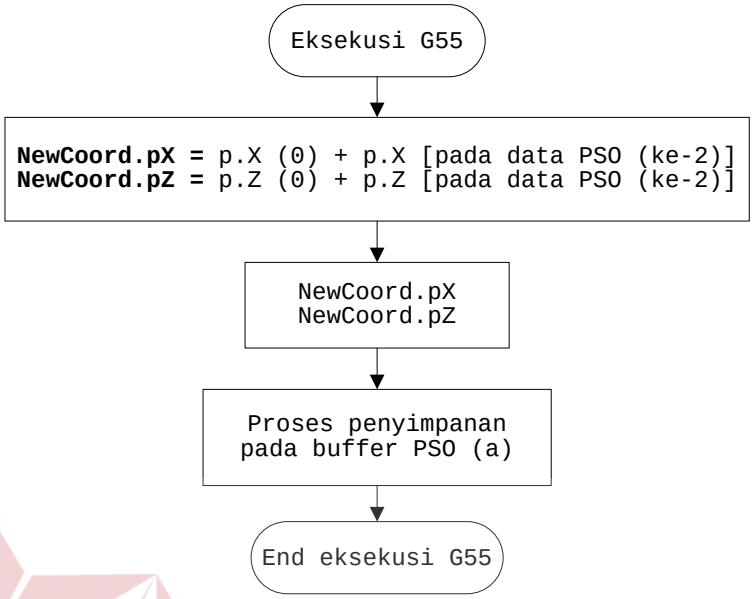
Dari bagan alir gambar 3.17 dapat diterangkan bahwa kode ini merupakan fungsi pergeseran titik nol dari titik nol mesin (0,0) ke posisi-1, dinamakan posisi-1 karena data (X,Z) berada pada PSO baris ke-1 yang merupakan data pergeseran milik G54. Koordinat baru dari pergeseran kode ini adalah penambahan dari (0.0) dengan data (X,Z) sub rutin PSO baris ke-1 dan *new Coord* hasil pergeseran ini disimpan pada *buffer* PSO (a) yang sudah disediakan.

Listing Program G54 :

```
void TAnimasiForm::G54_55()
{
    draft->noldraft= Point((draft->noldraft.x + g53.z*draft->pixel_per_cm/10), (draft->noldraft.y - g53.x*draft-
>pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x-g53.z/10;    //cm
    draft->nolchack.y= draft->nolchack.y-g53.x/10;   //cm
    draft->inc_point.x= draft->inc_point.x - g53.z;  //mm
    draft->inc_point.y= draft->inc_point.y - g53.x;  //mm
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle);
    //draft->DrawWorkpiece(BackCanvas);
}
```

B. Eksekusi Kode G55

Dari bagan alir gambar 3.18 dapat diterangkan bahwa kode ini merupakan fungsi pergeseran titik nol dari titik nol mesin (0,0) ke posisi-2, dinamakan posisi-2 karena data (X,Z) perintah pergeseran ini berada pada PSO baris ke-2 yang merupakan data pergeseran milik G55. Koordinat baru dari pergeseran kode ini adalah penambahan dari (0.0) dengan data (X,Z) subrutin PSO baris ke-2 dan *new Coord* hasil pergeseran ini disimpan pada *buffer* PSO (a) yang telah disediakan.



Gambar 3.18. *Flow chart* eksekusi kode G55

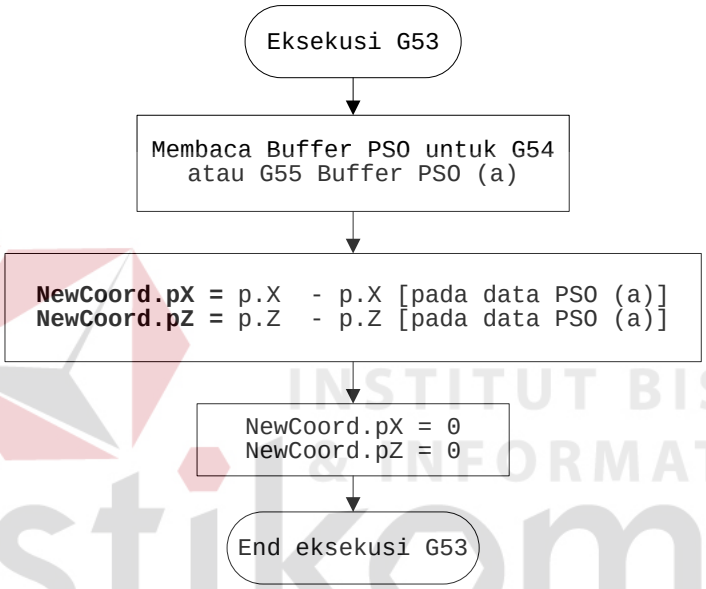
Listing Program G55 :

```
void TAnimasiForm::G54_55()
{
    draft->noldraft= Point((draft->noldraft.x + g53.z*draft->pixel_per_cm/10), (draft->noldraft.y - g53.x*draft->pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x-g53.z/10;    //cm
    draft->nolchack.y= draft->nolchack.y-g53.x/10;   //cm
    draft->inc_point.x= draft->inc_point.x - g53.z;  //mm
    draft->inc_point.y= draft->inc_point.y - g53.x;  //mm
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle);
    //draft->DrawWorkpiece(BackCanvas);
}
```

C. Eksekusi Kode G53

Fungsi dari kode G53 digambarkan dalam bentuk diagram alir gambar 3.19 Dari bagan alir tersebut kode G53 merupakan fungsi pembatalan pergeseran titik nol dari titik nol mesin. Pembatalan dari fungsi G54 dan G55, pada PSO baris ke-1 dan

ke-2 merupakan data pergeseran milik G54 dan G55. Subrutin dari PSO ini alan disimpan ke dalam *buffer* PSO (a), selanjutnya perintah G53 akan membaca *buffer* (a) sehingga dapat menentukan posisi nol akhir, kemudian melakukan pembatalan dengan harga sesuai yang tersimpan pada *buffer* PSO (a). Dengan proses ini referensi titik nol bergeser menuju titik nol mesin (M).



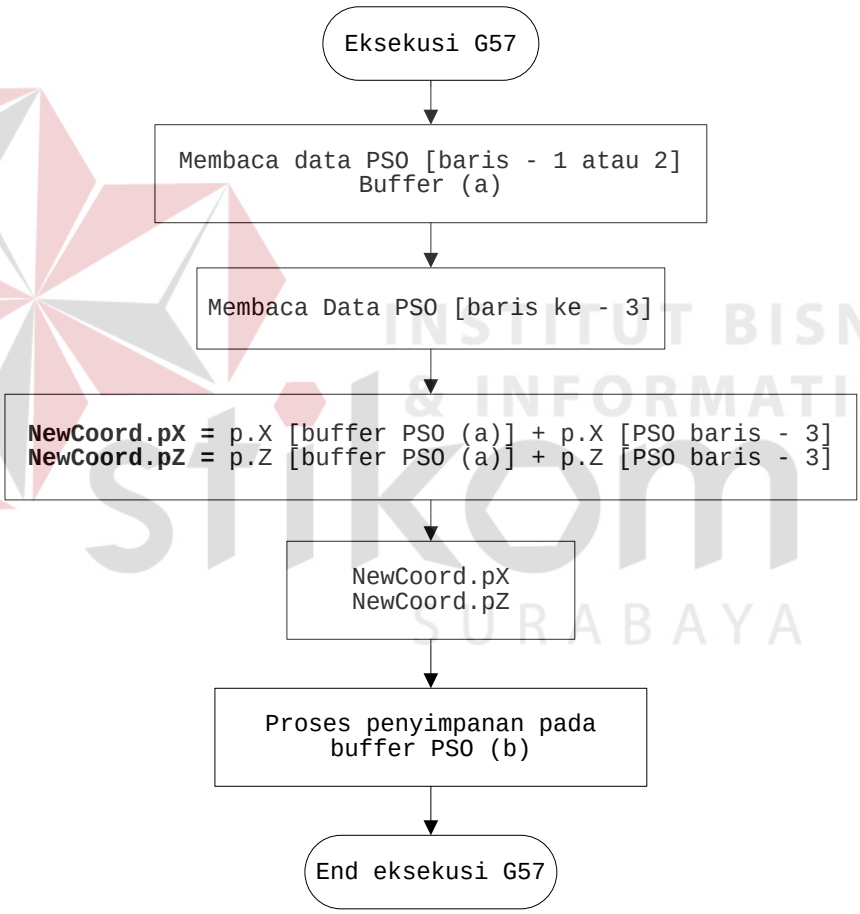
Gambar 3.19. *Flow chart* eksekusi kode G53

Listing Program G53 :

```
void TAnimasiForm::G53()
{
    draft->noldraft= Point((draft->noldraft.x - g53.z*draft->pixel_per_cm/10), (draft->noldraft.y + g53.x*draft-
>pixel_per_cm/10));
    draft->nolchack.x= (draft->nolchack.x + g53.z/10);   //cm
    draft->nolchack.y= (draft->nolchack.y + g53.x/10);   //cm
    draft->inc_point.x= draft->inc_point.x + g53.z;      //mm
    draft->inc_point.y= draft->inc_point.y + g53.x;      //mm
    g53.x=0; g53.z=0;
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle);
 }
```

D. Eksekusi Kode G57

Fungsi dari kode G57 ini merupakan fungsi pergeseran titik nol tidak dari titik nol mesin (0,0), tetapi pergeseran posisi titik nol referensi dari hasil pergeseran PSO beris ke-1 atau ke-2 [*buffer* PSO (a)]. Sedangkan data pegeseran untuk kelompok G57, G58, dan G59 pada *software* ini diberikan fasilitas penyimpan, yaitu di *buffer* PSO (b), pada diagram ini dapat dilihat proses dari G57 :



Gambar 3.20. *Flow chart* eksekusi kode G57

Liting Program G57 :

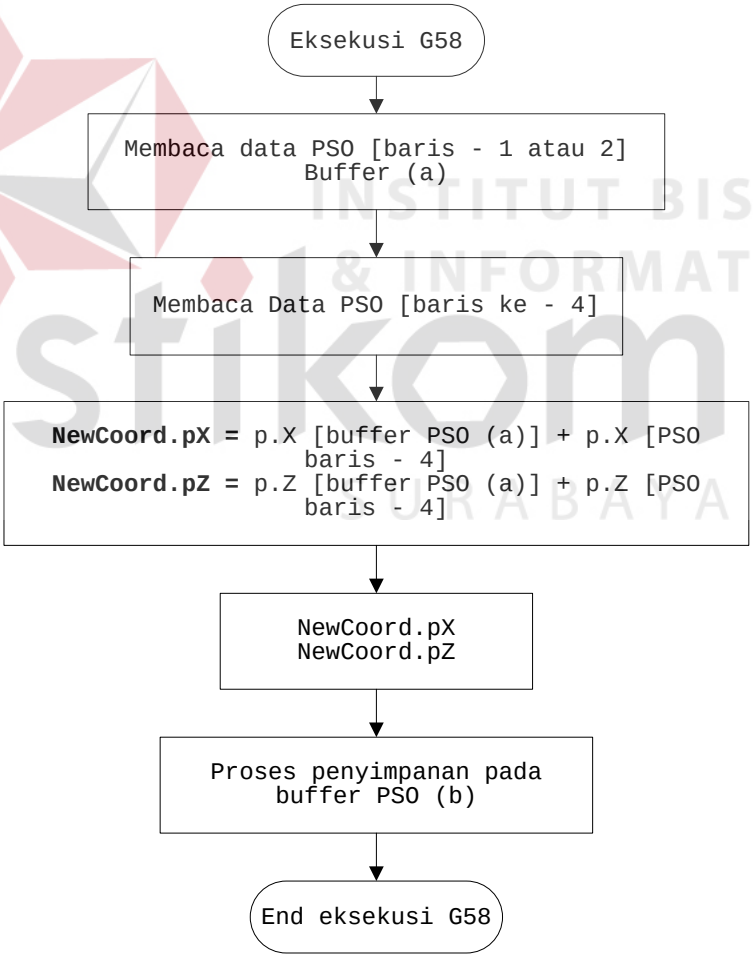
```
void TAnimasiForm::G57 ()
{
    draft->noldraft= Point((draft->noldraft.x + g56.z*draft->pixel_per_cm/10), (draft->noldraft.y - g56.x*draft-
>pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x-g56.z/10;  //cm
    draft->nolchack.y= draft->nolchack.y-g56.x/10;  //cm
    draft->inc_point.x= draft->inc_point.x - g56.z; //mm
    draft->inc_point.y= draft->inc_point.y - g56.x; //mm
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle );
    }
```

Dengan diagram alir gambar 3.20 dapat dilihat bahwa fungsi G57 adalah fungsi pergeseran dari hasil pergeseran referensi titik nol dari titik nol mesin yaitu kode G54 atau G55 [*buffer* PSO (a)]. Setelah ada perintah G57 ini, maka modul tersebut membaca kondisi titik nol referensi yang ada pada *buffer* PSO (a). Sedangkan hasil pergeseran titik referensi titik nol ini merupakan penambahan dari kondisi terakhir (*buffer* PSO a) dengan data PSO baris ke-3, sehingga menghasilkan koordinat referensi baru. Setelah koordinat baru tercapai, maka kondisi tersebut disimpan pada *buffer* PSO (b). Pengelompokan *buffer* PSO ini bertujuan sebagai inisialisasi posisi titik referensi dalam hubungannya dengan fungsi pembatalan setiap kelompok PSO itu sendiri.

E. Eksekusi Kode G58

Fungsi dari kode G58 sama dengan fungsi pergeseran titik nol referensi G57 di atas tidak dari titik nol mesin (0,0). Sedangkan data pergeseran untuk fungsi G58 ini berada pada PSO baris ke-4 [*buffer* PSO (b)].

Yang membedakan antara G58 dengan G57 adalah setelah membaca kondisi titik nol referensi yang ada pada *buffer* PSO (a), maka fungsi G58 ini menterjemahkan data PSO baris ke-4, yang merupakan data milik fungsi G58 itu sendiri. Sedangkan hasil pergeseran titik referensi dari titik nol ini penambahan dari kondisi teakhir (*buffer* PSO a) dengan data PSO baris ke-4, sehingga menghasilkan koordinat referensi baru. Setelah koordinat baru tercapai maka kondisi ini disimpan pada *buffer* PSO (b). Pada diagram alir di bawah ini dapat dilihat proses dari G58 :



Gambar 3.21. *Flow chart* eksekusi kode G58

Liting Program G58 :

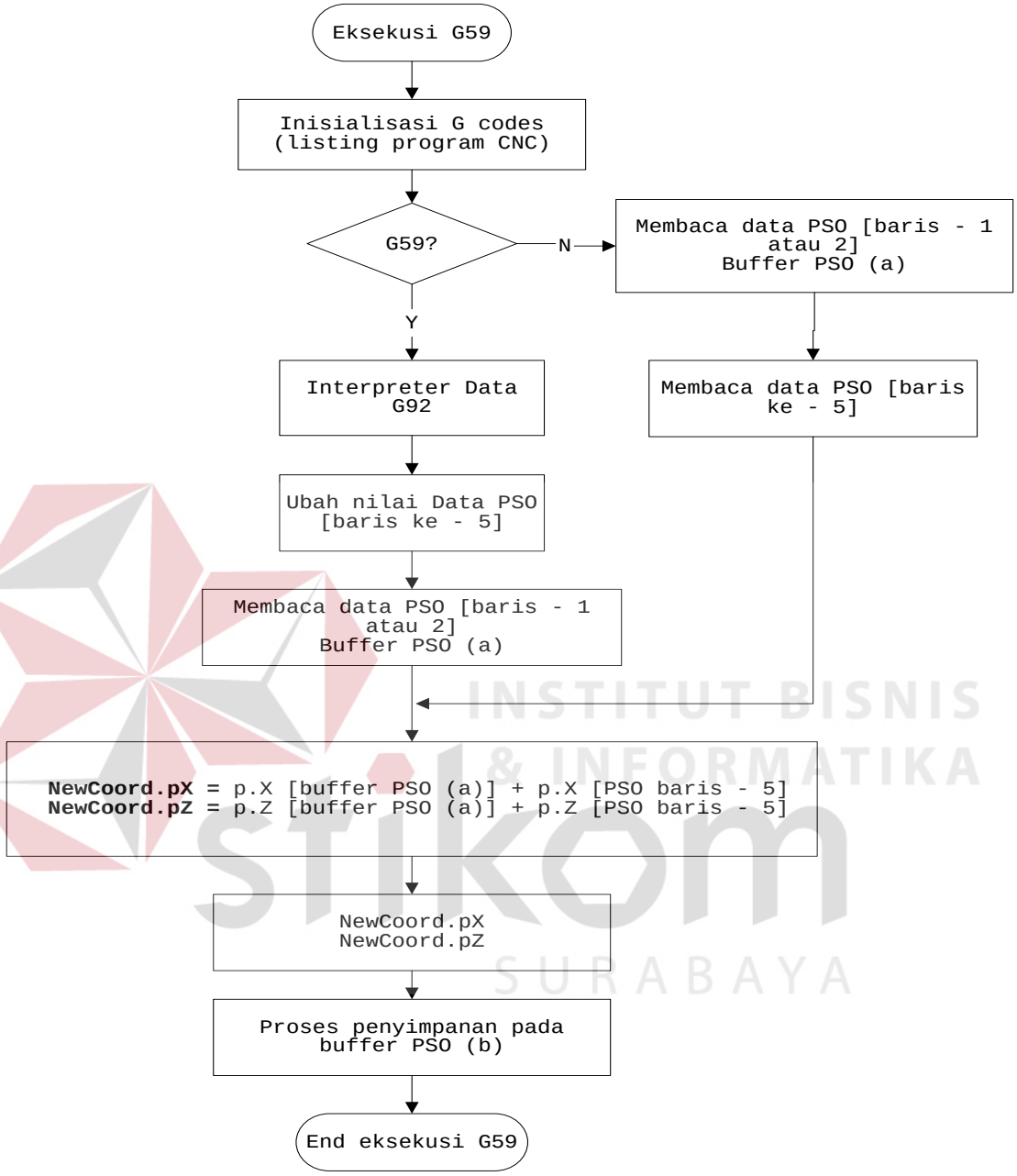
```
void TAnimasiForm::58 ()
{
    draft->noldraft= Point((draft->noldraft.x + g56.z*draft->pixel_per_cm/10), (draft->noldraft.y -
g56.x*draft->pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x-g56.z/10;  //cm
    draft->nolchack.y= draft->nolchack.y-g56.x/10;  //cm
    draft->inc_point.x= draft->inc_point.x - g56.z; //mm
    draft->inc_point.y= draft->inc_point.y - g56.x; //mm
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle );
    //draft->DrawWorkpiece(BahanCanvas);
}
```

F. Eksekusi Kode G59

Kode G59 ini memiliki dua fungsi pada implementasinya dipemrograman mesin pekakas CNC. Fungsi dari kode G59 adalah :

1. Untuk pergeseran titik nol referensi nol mesin, dalam hal ini sama dengan fungsi kode-kode G57 dan G58.
2. Untuk pengaktif kode G92, yang merupakan pergeseran referensi tidak dari titik nol mesin dan menunjukkan perintah, program dijalankan secara absolut, dari nilai X dan Z yang dimasukkan.

Dengan adanya pengaktifan G59 ini dalam *listing* pemrograman CNC, maka secara langsung pada pencatat PSO baris ke-5 nilainya diganti sesuai dengan nilai data (X,Z) hasil dari masukan kode G92, ini berlaku bila diaktifkan dengan fungsi kode G59.



Gambar 3.22. *Flow chart* eksekusi kode G59

Dari diagram alir gambar 3.22 dapat dilihat bahwa dengan adanya perintah dari kode G92 maka secara langsung data dari pencatat PSO baris ke-5 berubah nilainya sesuai dengan harga X, Z nya, data yang diaktifkan oleh kode ini disimpan di

sub rutin PSO baris untuk G59 (ke-5) untuk menjalankan keseluruhan proses pemesinan secara absolut dari titik referensi tersebut.

Liting Program G59 :

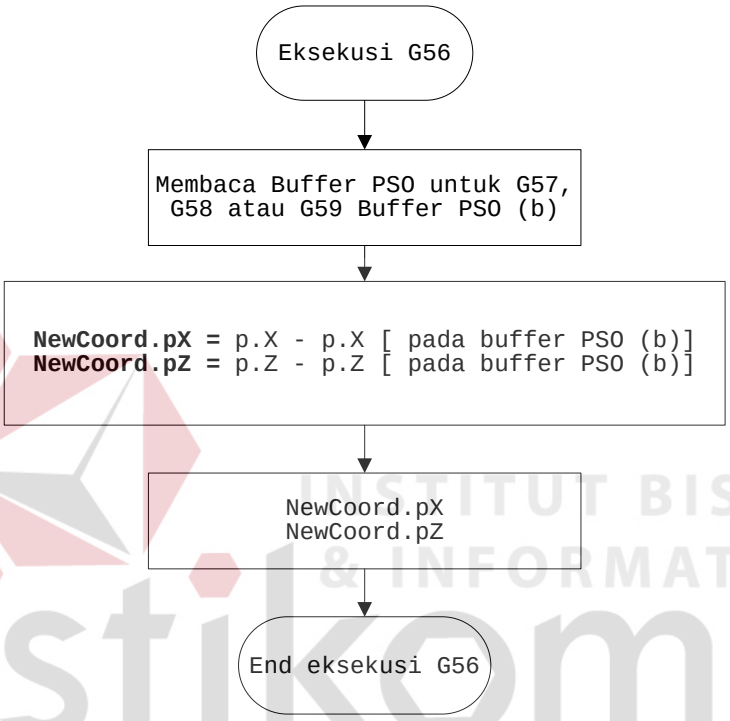
```
void TAnimasiForm:: 59()
{
    draft->noldraft= Point((draft->noldraft.x + g56.z*draft->pixel_per_cm/10), (draft->noldraft.y -
g56.x*draft->pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x-g56.z/10;  //cm
    draft->nolchack.y= draft->nolchack.y-g56.x/10;  //cm
    draft->inc_point.x= draft->inc_point.x - g56.z; //mm
    draft->inc_point.y= draft->inc_point.y - g56.x; //mm
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle );
    //draft->DrawWorkpiece(BahanCanvas);
}
```

Liting Program G92 :

```
void TAnimasiForm::g92Scan(AnsiString temp[])
{
  for(int i=1; i<3; i++)
   {
    if(temp[i].SubString(1,1)=="X")
     {
      temp[i]= temp[i].Delete(1,1);
      g92.x= temp[i].ToDouble();
     }
    else if(temp[i].SubString(1,1)=="Z")
    {
      temp[i]= temp[i].Delete(1,1);
      g92.z= temp[i].ToDouble();
    }
   }
}
```

G. Eksekusi Kode G56

Fungsi dari kode G56 adalah kode fungsi untuk pembatalan dari hasil pergeseran dari kode G57, G58, dan G59.



Gambar 3.23. *Flow chart* eksekusi kode G56

Dari gambar 3.23, bagan alir tersebut dapat menerangkan fungsi pembatalan pergeseran titik nol tidak dari titik nol mesin yang disimpan pada *buffer* PSO (b). perintah G56 ini akan membasa *buffer* (b) sehingga dapat menentukan posisi nol akhir, kemudian melakukan pembatalan dengan harga yang tersimpan pada *buffer* PSO (b). dengan proses ini maka referensi titik nol bergeser menuju titik nol mesin senilai harga pada pencatat PSO baris ke-3, 4 atau 5.

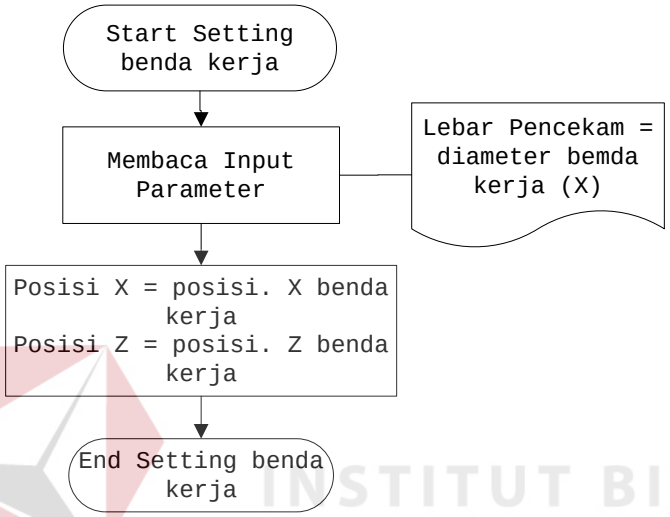
Liting Program G56 :

```
void TAnimasiForm::G56()
{
    draft->noldraft= Point((draft->noldraft.x - g56.z*draft->pixel_per_cm/10), (draft->noldraft.y +
g56.x*draft->pixel_per_cm/10));
    draft->nolchack.x= draft->nolchack.x+g56.z/10;  //cm
    draft->nolchack.y= draft->nolchack.y+g56.x/10;  //cm
    draft->inc_point.x= draft->inc_point.x + g56.z; //mm
    draft->inc_point.y= draft->inc_point.y + g56.x; //mm
    g56.x=0; g56.z=0;
    BackCanvas->Brush->Color=syscol;
    BackCanvas->FillRect(Rect_Back);
    draft->DrawMiliMeter(BackCanvas);
    draft->DrawChack(BackCanvas, data[0].spindle);
    //draft->DrawWorkpiece(BahanCanvas);
}
```

3.1.3. Perancangan Modul Benda Kerja

Pada software simulasi ini kami tidak membahas program *polygon* kami hanya menyediakan satu material benda kerja, akan tetapi dapat diatur dimensinya secara manual. Pengaturan dimensi benda kerja dilakukan pada mode *automatic* ketika akan melakukan proses simulasi pada *software* simulasi ini. Pada *emco draft* pemilihan program *polygon* adalah sebelum kita mengaktifkan proses dengan panel *Cycle Start*, disana ada perintah pemilihan program *polygon* nomor berapa yang kita pilih. Sedangkan pada software ini kita dapat menyesuaikan dimensi benda kerja dengan program G *codes* yang sudah disusun dan siap untuk disimulasikan prosesnya. Modul benda kerja pada *software* ini terdapat dalam *form* animasi. *Form* animasi ini memuat semua modul yang berkaitan dengan jalannya proses simulasi (animasi).

Dari diagram alir gambar 3.24 dapat dijelaskan bahwa data inputan dari *setting* benda kerja adalah X dan Z, dimana X merupakan diameter benda kerja dan Z adalah panjang dari benda kerja. Titik (0,0) dari benda kerja untuk memulai *polygon* adalah ujung kiri atas benda kerja berhimpit dengan pencekam mesin CNC.



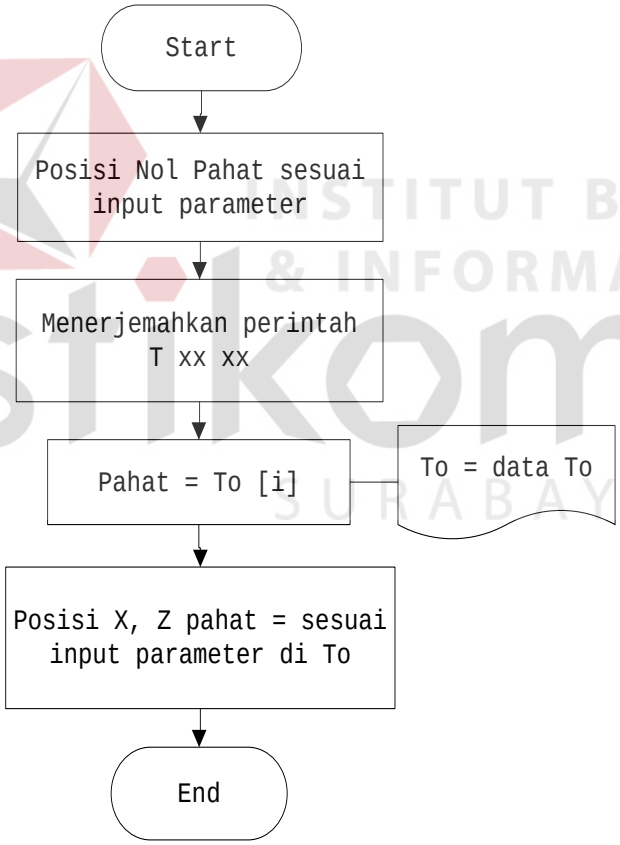
Gambar 3.24. *Flow chart* eksekusi *setting* benda kerja

Listing Program setting benda kerja :

```
void __fastcall TAnimasiForm::meDiameterKeyPress(TObject *Sender, char &Key)
{
if(Key==VK_RETURN)
{
 draft->material_dia= meDiameter->Text.ToDouble()/10.0;
 draft->material_length= mePanjang->Text.ToDouble()/10.0;
 BackCanvas->Brush->Color=syscol;
 BackCanvas->FillRect(Rect_Back);
 draft->DrawMiliMeter(BackCanvas);
 draft->DrawChack(BackCanvas, data[0].spindle);
 WorkCanvas->CopyRect(Rect_Back,BackCanvas,Rect_Back);
 draft->DrawWorkpiece(BahanCanvas);
 draft->MakePolygon(BahanCanvas, WorkCanvas);
 //draft->DrawPoligon(BahanCanvas, WorkCanvas);
 tool=0;
 draft->DrawPahat(WorkCanvas, BahanCanvas,draft->inc_point.x, draft->inc_point.y, tool);
 //aft->MakePolygon(BahanCanvas, WorkCanvas);
 //draft->DrawPoligon(BahanCanvas, WorkCanvas);
 PB->Canvas->CopyRect(Rect_Back,WorkCanvas,Rect_Back);
}
}
```

3.1.4. Perancangan Modul *tool library*

Pada modul ini kami sediakan beberapa *tool* yang dapat dipilih dan dapat dimasukkan nomer koreksinya kedalam data TO. Pada pembahasan *software* ini data TO merupakan subrutin yang memiliki *file* memori, sehingga pada saat proses simulasi berlangsung modul dari *tool* ini dapat memanggil data koreksinya pada data TO dan menterjemahkan dari perintah–perintah pemrograman G *codes* dalam pergantian *tool*.



Gambar 3.25. *Flow chart* eksekusi *setting tool*

Untuk pergantian pahat dapat dilihat pada diagram alir gambar 3.25, pada saat pembacaan G *code* pahat akan berjalan ke posisi sesuai dengan nilai parameter data X,Z, setelah ada perintah pergantian pahat (Txx xx) maka akan diterjemahkan sesuai dengan data To[i] yang berisi data konpensasi pahat yang berada di subrutin TO pada *software* ini. Setelah itu pahat melakukan pergantian sesuai dengan data parameter G *code* pada blok program sebelumnya, untuk *setting* pahat yang akan diletakkan pada *turet* pahat dapat dilakukan dengan mekanisme pemberian nomor sesuai dengan nomor *turet* pada *tool display*

Tabel 3.2. Data *too*l

Nomer *Tool*	***Bentul Tool***
T0101	
T0202	
T0303	
T0404	
T0505	
T0606	

Listing Program Setting tool :

```
void __fastcall TDRAFT::DrawPahat(TCanvas *can, TCanvas *candes, double zp, double xp, int tool)
{
 //save canvas
 TCanvas *oldcanvas = new TCanvas;
 oldcanvas->Brush->Style = can->Brush->Style;
 oldcanvas->Brush->Color = can->Brush->Color;
 oldcanvas->Pen->Color = can->Pen->Color;
 //set new can properties
```

```
can->Pen->Color= 0x009D6A00;
can->Pen->Style= psSolid;
can->Brush->Color=clRed;
can->Brush->Style=bsSolid;
//-----------
zp=zp/10.0;
xp=xp/10.0;
int zo= pixel_per_cm;
//-----------
int a,b;
switch(tool)
{
  case 1:   a=1; b=4; break;
  case 2:   a=7; b=0; break;
  case 3:   a=1; b=4; break;
  case 4:   a=7; b=0;break;
  case 5:   a=1; b=4;break;
  case 6:   a=7; b=-0.5;break;
  default:  a=1; b=4; break;
}
int z= noldraft.x + (zp+a)*zo;
int x= noldraft.y - (xp+b)*zo;
//-----------
TPoint base_pahat1[]={
        Point(z+6*zo, x-11*zo),
        Point(z+14*zo, x-11*zo),
        Point(z+14*zo, x-1*zo),
        Point(z+6*zo, x-1*zo),
        Point(z+6*zo, x-11*zo),
};

TPoint base_pahat5[]={
        Point(z-2*zo,  x-1*zo),
        Point(z, x-1*zo),
        Point(z, x+1*zo),
        Point(z-2*zo,  x+1*zo),
        Point(z-2*zo, x-1*zo),
};

//======pahat area=========

 z= noldraft.x + (zp)*zo;
 x= noldraft.y - (xp)*zo;
 TPoint mid_pahat1[]={
        Point(z+0.04*zo, x-0.6*zo),
        Point(z+0.25*zo, x-1*zo),
        Point(z+0.25*zo, x-6*zo),
        Point(z+1.25*zo, x-6*zo),
        Point(z+1.25*zo, x-0.8*zo),
        Point(z+1*zo,    x-0.6*zo),
        Point(z+0.04*zo, x-0.6*zo),
};
};
TPoint pahat2[]={  //center drill
        Point(z+1.2*zo, x),
        Point(z+1.2*zo, x+0.4*zo),
        Point(z+1*zo,   x+0.4*zo),
        Point(z+0.6*zo, x+0.2*zo),
        Point(z+0.1*zo, x+0.2*zo),
        Point(z,        x),
```

```
        Point(z+0.1*zo, x-0.2*zo),
        Point(z+0.6*zo, x-0.2*zo),
        Point(z+1*zo,   x-0.4*zo),
        Point(z+1.2*zo, x-0.4*zo),
};

 TPoint pahat6[]={
        Point(z,  x),
        Point(z+0.3*zo,  x+0.15*zo),
        Point(z+0.5*zo,  x+0.5*zo),
        Point(z+0.15*zo, x+0.3*zo),
};


can->Polyline(base_pahat1,4);
can->Polyline(base_pahat2,4);
can->Polyline(base_pahat3,4);
can->Polyline(base_pahat4,4);
can->Polyline(base_pahat5,4);
//============================

switch(tool)
{
  case 1:  can->Polyline(mid_pahat1,6); can->Polygon(pahat1,9);  break;
  case 2:  can->Polyline(mid_pahat2,4); can->Polygon(pahat2,9);  break;
  case 3:  can->Polyline(mid_pahat3,6); can->Polygon(pahat3,11); break;
  case 4:  can->Polyline(mid_pahat4,4); can->Polygon(pahat4,7); break;
  case 5:  can->Polyline(mid_pahat5,4); can->Polygon(pahat5,3); break;
  case 6:  can->Polyline(mid_pahat6,5); can->Polygon(pahat6,3); break;
}
//============================
//============================
candes->Pen->Color= clWhite;
candes->Pen->Style=psSolid;
candes->Brush->Color=clWhite;
candes->Brush->Style=bsSolid;
switch(tool)
{
  case 1:   candes->Polygon(pahat1,9); break;
  case 2:   candes->Polygon(pahat2,9); break;
  case 3:   candes->Polygon(pahat3,11); break;
  case 4:   candes->Polygon(pahat4,7);break;
  case 5:   candes->Polygon(pahat5,3);break;
  case 6:   candes->Polygon(pahat6,3);break;
}
can->Pen->Style= psClear;
can->Brush->Color=clNavy;
can->Brush->Style=bsSolid;
double r=0.25;
zo= pixel_per_cm;
z= noldraft.x + (zp+a)*zo;
x= noldraft.y - (xp+b)*zo;
can->Pie((z-r*zo), (x-r*zo), (z+r*zo), (x+r*zo), (z), (x-r*zo), (z), (x-r*zo));
can->Brush->Color=clWhite;
can->Pie((z-r*zo), (x-r*zo), (z+r*zo), (x+r*zo), (z+r*zo), (x-r*zo), (z-r*zo), (x-r*zo));
can->Pie((z-r*zo), (x-r*zo), (z+r*zo), (x+r*zo), (z-r*zo), (x+r*zo), (z+r*zo), (x+r*zo));


//store canvas
can->Brush->Style= oldcanvas->Brush->Style;
```

```
 can->Brush->Color= oldcanvas->Brush->Color;
 can->Pen->Color=   oldcanvas->Pen->Color;
 delete oldcanvas;
}
```

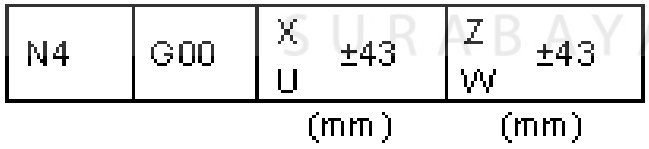
3.1.5. Perancangan Modul setiap fungsi G *codes* dan M *codes*

Modul G *codes* dan M *codes* dalam *software* ini kami masukkan dalam subrutin pada *form* utama animasi, setiap subrutin tersebut berisi modul yang dapat membaca segala perintah gerakan dari pemrograman G *codes* dengan menterjemahkan semua parameter–parameter dari data *input*.

Berikut akan dijelaskan beberapa perintah kode G dan M yang digunakan pada program ini.

G00 (Gerakan Cepat)

G00 (gerakan cepat) adalah gerakan jalan murni tanpa pengerjaan, kecepatan dari gerak cepat ditentukan oleh pabrik untuk setiap jenis mesin.



Gambar 3.26. Penulisan program G00.

Keterangan :

1) N : Nomor blok

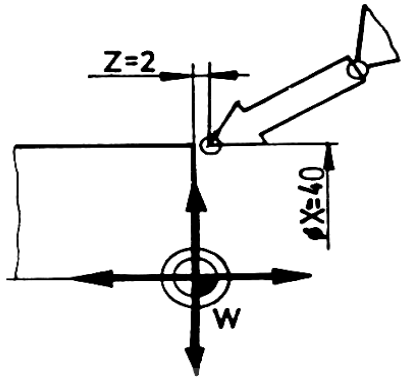
2) G00 : Gerakan cepat

3) X,Z : Koordinat tujuan dalam absolut

4) U,W : Koordinat tujuan dalam inkremental

Contoh G code dari gambar 3.27 :

N100….
N110 G00 X40.000 Z2.000
N120…..



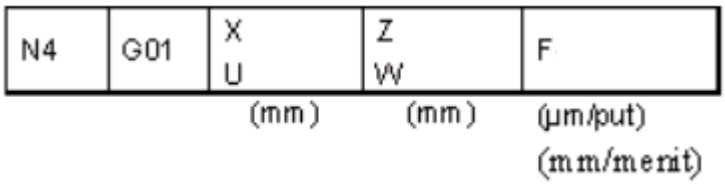
Gambar 3. 27. Contoh G00 gerak cepat

Listing Program G00 :

```
void __fastcall TDRAFT::G00(double z, double x, bool absolut){
  finish=false;
  double X,Y;
  speed=2000;
  if(absolut)
  {
   X= (abs_point.x+z);
   Y= (abs_point.y+ double(x/2));
  }
  else
  {
   X= (inc_point.x+z);
   Y= (inc_point.y+x);
  }
  length_point.y=Y-inc_point.y;
  length_point.x=X-inc_point.x;

  radius= sqrt(pow(length_point.x, 2) + pow(length_point.y, 2));
  if(radius!=0)
  {
   COS= length_point.x/radius;
   SIN= length_point.y/radius;
  }
  else finish=true;
}
```

G01 (Interpolasi Lurus)

G01 adalah gerakan pengerjaan lurus, *feeding* harus diprogram, dapat dalam mm/menit (G94) atau dalam µm/put (G95).



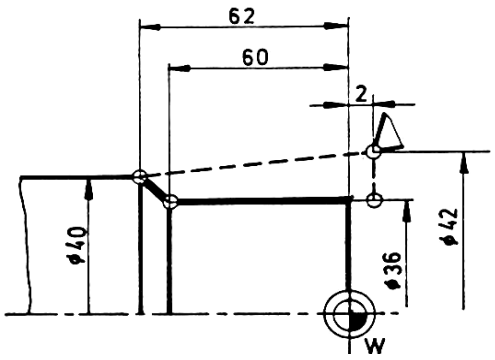
Gambar 3.28. Penulisan program G01

Keterangan :

1) N : Nomor blok
2) G01 : Interpolasi lurus
3) X,Z : Koordinat tujuan dalam absolut
4) U,W : Koordinat tujuan dalam inkremental
5) F : *Feeding*

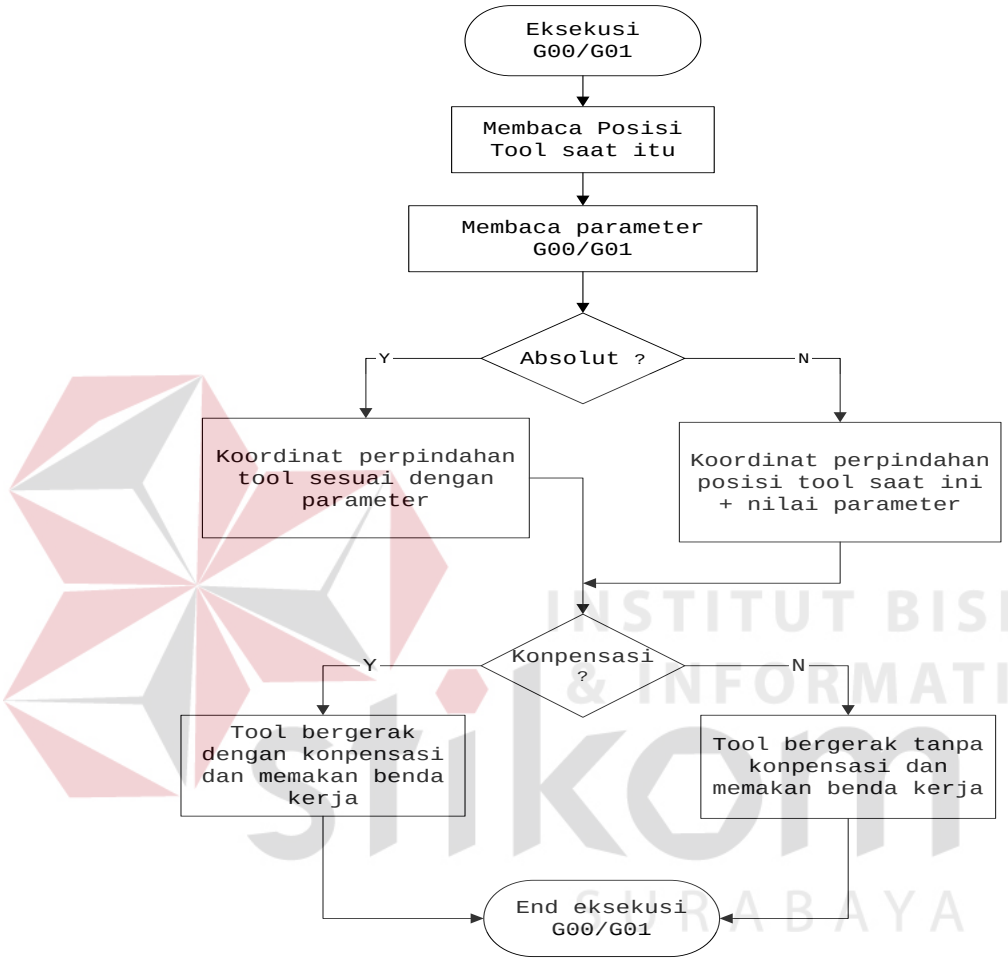
Contoh program G code dari gambar 3.29:

```
N110 G00 X42 Z2.00
N120 G00 X36.000
N130 G01 Z-60.000
N140 G01 X40.000 Z-62.000
N150 G00 X42.000 Z2.000
```



Gambar 3.29. Contoh Interpolasi lurus

Pada pembahasan dalam *software* ini eksekusi G00 dan G01 dirancang seperti yang dijelaskan pada diagram alir di bawah ini :



Gambar 3.30. *Flow chart* eksekusi kode G00/G01

Dari bagan alir gambar 3.30 dapat dijelaskan bahwa, untuk mengeksekusi kode G01/02 terlebih dahulu menentukan dan membaca posisi *tool* saat itu, kemudian membaca parameter kodenya apakah *absolute* atau *incremental*. Jika program dijalankan secara absolut maka tool akan bergerak dengan koordinat berpindah sesuai dengan parameter, bila *incremental* maka tool bergerak dengan koordinat

perpindahan posisi *tool* sekarang ditambah dengan nilai parameter. pada proses gerakan ini *tool* akan dihadapkan pada kondisi terkompensasi atau tidak.

Listing Program G01 :

```
void __fastcall TDRAFT::G01(double z, double x, bool absolut, int f)
{
  finish=false;
  double X,Y;
  speed= f;
  if(absolut)
  {
    X= (abs_point.x+z);
    Y= (abs_point.y+ double(x/2));
  }
  else
  {
    X= (inc_point.x+z);
    Y= (inc_point.y+x);
  }
  length_point.y=Y-inc_point.y;
  length_point.x=X-inc_point.x;

  radius= sqrt(pow(length_point.x, 2) + pow(length_point.y, 2));
  if(radius!=0)
  {
    COS= length_point.x/radius;
    SIN= length_point.y/radius;
  }
  else finish=true;

}
```


G02-G03 (Interpolasi Melingkar)

G02 Interpolasi melingkar searah jarum jam. G03 Interpolasi melingkar berlawanan arah jarum jam

N4	G02 G03	X U	Z W	I	K	F
		(mm)	(mm)	(mm)	(mm)	(µm/put)

Gambar 3.31. Penulisan program G01/G02

Keterangan :

1) N : Nomor blok

1) G02 : Interpolasi melingkar dalam arah jarum jam

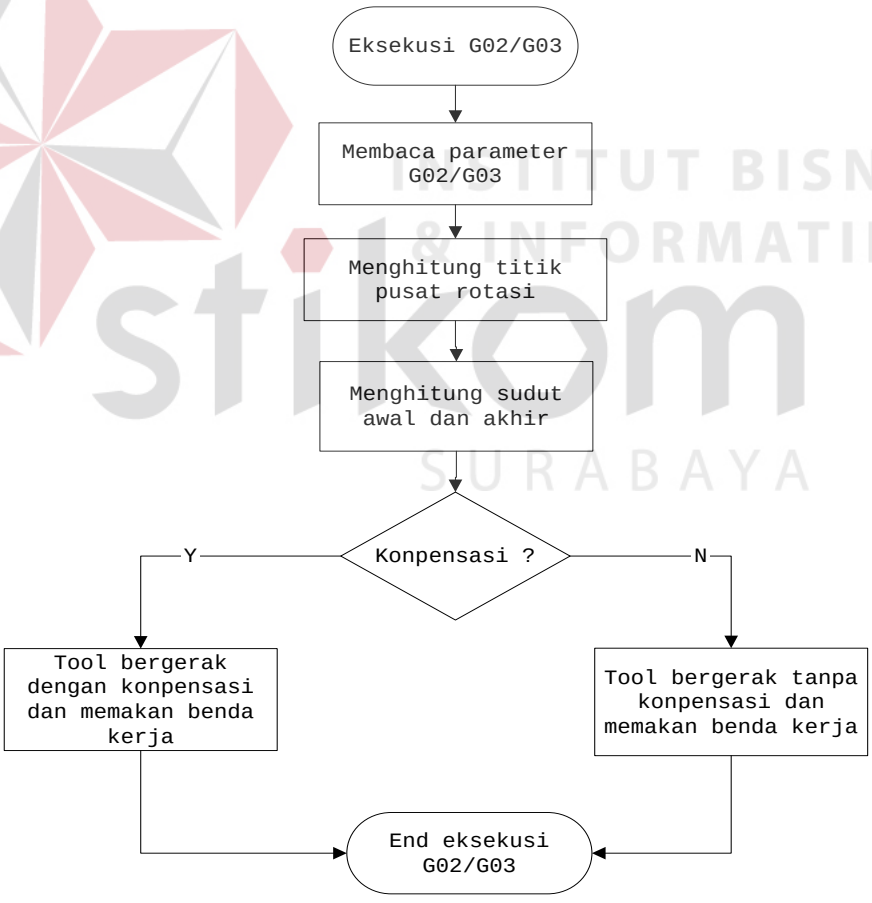
2) G03 : Interpolasi melingkar dalam arah berlawanan arah jarum jam

3) X,Z : Koordinat tujuan dalam absolut

4) U,W : Koordinat tujuan dalam inkremental

5) I,K : Jarak titik pusat lingkaran dari titik awal lingkaran

6) F : *Feeding*

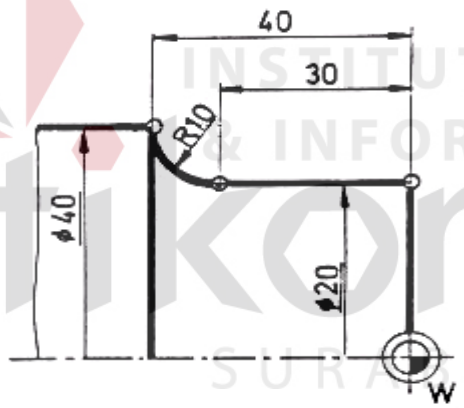


Gambar 3.32. *Flow chart* G02/G03

Bagan alir pada gambar 3.32. menerangkan proses eksekusi dari G02 dan G03 akan dilakukan dengan membaca parameter dari kode, kemudian menentukan titik pusat dan menghitung sudut awal dan sudut akhir rotasi, kemudian apakah kondisi dari gerakan tersebut berkompensasi atau tidak. Bila tidak, *tool* akan bergerak pada lintasan tanpa kompensasi atau sebaliknya, *tool* akan bergerak dengan terkompensasi.

Contoh program G code dari gambar 3.33 :

```
N100…..
N110 G01 X20.000 Z-30.000
N120 G02 X40.000 Z-40.000 I10.000 K00.000
N130…..
```



Gambar 3.33. G02 Interpolasi melingkar

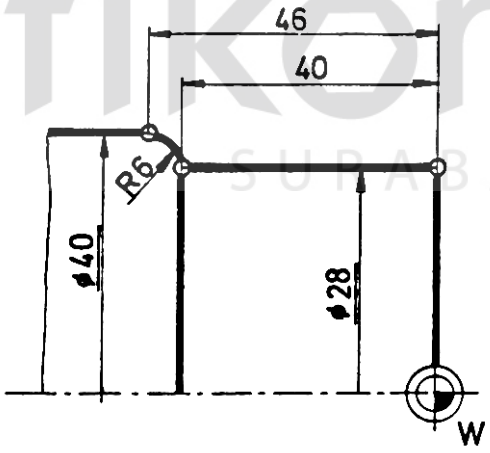
Listing Program G02 :

```
void __fastcall TDRAFT::G02(double z, double x, double i, double k, bool absolut, int f)
{
double r,a,b,deg1,deg2;
double X,Y;
  finish=false;
  speed= f;
  if(absolut)
  {
    X= (abs_point.x+z);
```

```
    Y= (abs_point.y+ double(x/2));
  }
  else
  {
    X= (inc_point.x+z);
    Y= (inc_point.y+x);
  }

a= inc_point.x+k;
b= inc_point.y+i;

r= sqrt(pow((X-a),2) + pow((Y-b),2));
if(fabs(X-a)!=0)  deg1= 360*acos(fabs(X-a)/r)/(2*M_PI);
if(fabs(Y-b)!=0)  deg1= 360*asin(fabs(Y-b)/r)/(2*M_PI);

r= sqrt(pow((inc_point.x-a),2) + pow((inc_point.y-b),2));
if(fabs(inc_point.x-a)!=0)  deg2= 360*acos(fabs(inc_point.x-a)/r)/(2*M_PI);
if(fabs(inc_point.y-b)!=0)  deg2= 360*asin(fabs(inc_point.y-b)/r)/(2*M_PI);
degre= (180-(deg1+deg2));
I=i;
K=k;
CTRA=a;
CTRB=b;
```

Contoh program G code dari gambar 3.34 :

```
N100…..
N110 G01 X28.000 Z-40.000
N120 G03 X40.000 Z-46.000 I00.000 K-6.000
N130…..
```



Gambar 3.34. G02 Interpolasi melingkar

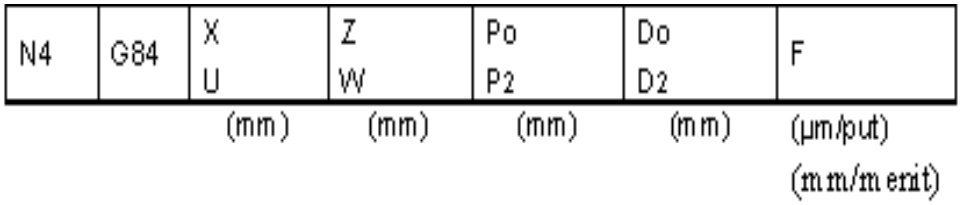
Listing Program G03 :

```
void __fastcall TDRAFT::G03(double z, double x, double i, double k, bool absolut, int f)
{
```

```
double r,a,b,deg1,deg2;
double X,Y;
  finish=false;
  speed= f;
  if(absolut)
  {
    X= (abs_point.x+z);
    Y= (abs_point.y+ double(x/2));
    //ShowMessage(AnsiString(X)+", "+AnsiString(Y));
  }
  else
  {
    X= (inc_point.x+z);
    Y= (inc_point.y+x);
  }

a= inc_point.x+k;
b= inc_point.y+i;
//ShowMessage(AnsiString(a)+", "+AnsiString(b));
r= sqrt(pow((X-a),2) + pow((Y-b),2));
if(fabs(X-a)!=0)  deg1= 360*acos(fabs(X-a)/r)/(2*M_PI);
if(fabs(Y-b)!=0)  deg1= 360*asin(fabs(Y-b)/r)/(2*M_PI);

r= sqrt(pow((inc_point.x-a),2) + pow((inc_point.y-b),2));
if(fabs(inc_point.x-a)!=0)  deg2= 360*acos(fabs(inc_point.x-a)/r)/(2*M_PI);
if(fabs(inc_point.y-b)!=0)  deg2= 360*asin(fabs(inc_point.y-b)/r)/(2*M_PI);
degre= -(180-(deg1+deg2));
I=i;
K=k;
CTRA=a;
CTRB=b;
 }
```

G84 (Siklus Pembubutan)

Siklus pembubutan memanjang, pada alir diagram dibawah ini menjelaskan eksekusi fungsi dari G84 :



Gambar 3.35. Penulisan program G84

Keterangan :

1) N : Nomor blok

2) G84 : Siklus pembubutan memanjang

3) X,Z : Koordinat tujuan dalam absolut

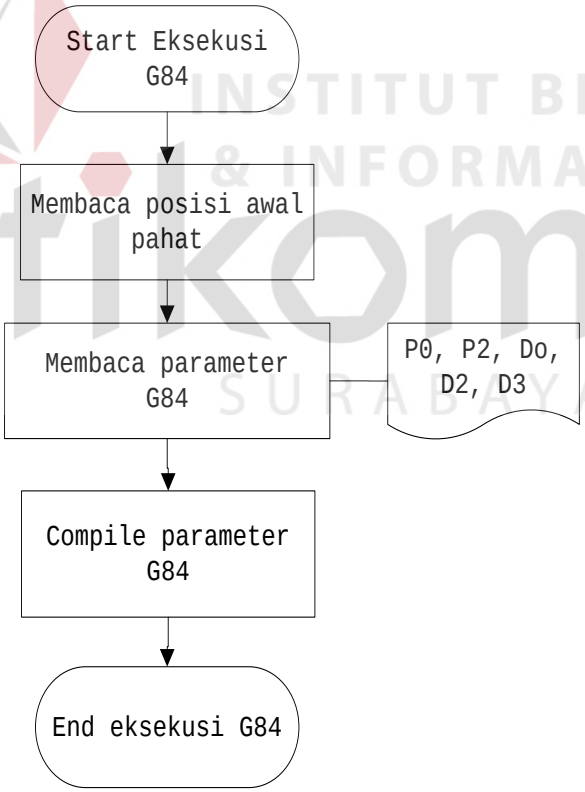
4) U,W : Koordinat tujuan dalam inkremental

5) P_0/P_2 : Ukuran tirus (absolut atau inkremental)

6) D_0/D_2 : Kelebihan ukuran (absolut atau inkremental)

7) D_3 : Pembagian dalam pemotongan

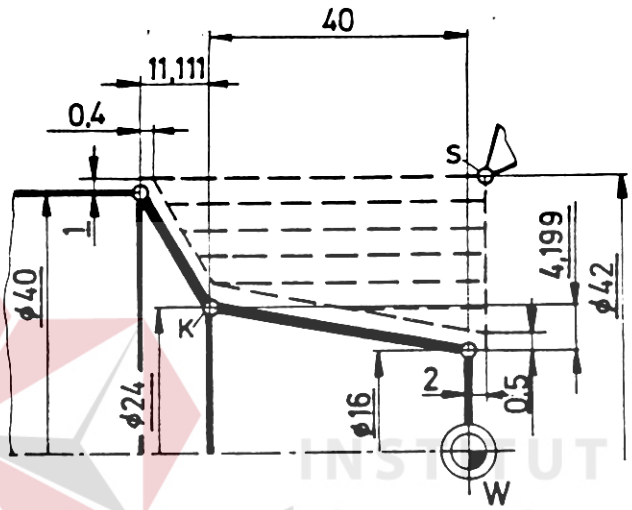
8) F : *Feeding*



Gambar 3.36. *Flow chart* kode G84

Contoh program G code dari gambar 3.37 :

N100 …….
N110 G00 X42.000 Z2.000
N120 G84 X24.000 Z-40.000 /P_0-4.199/P2-11.111
N130 ……..



Gambar 3.37. Contoh G84 siklus pembubutan mamanjang

Listing Program G84 :

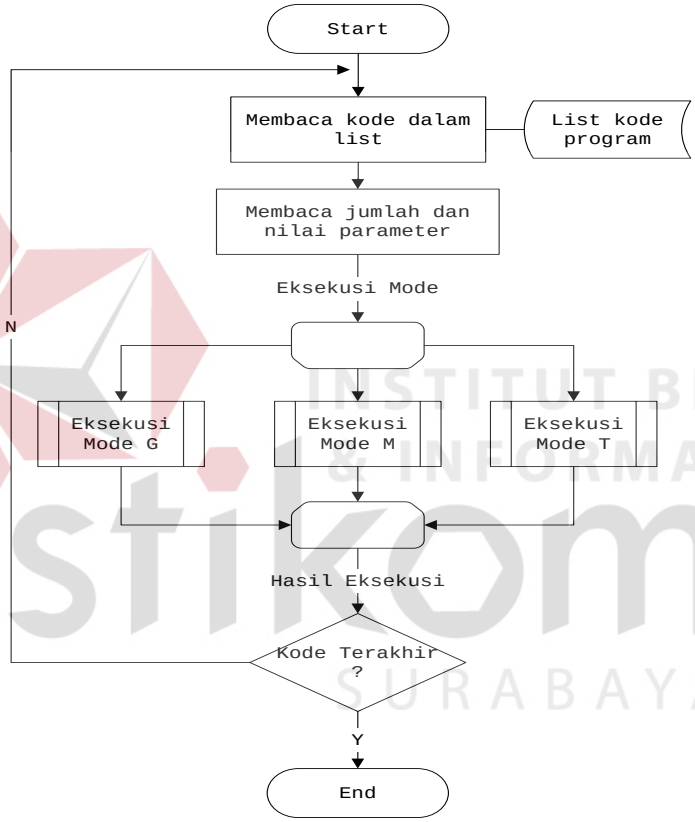
```
void TGCODE ::getpos84(TRichEdit *mDest,int F)
{
 double x0,y0,x1,y1,x2,y2,x3,y3,d0,d2,d3,p0,p2,X,Y,xa,ya;
 d0=Pan[0].l/1000.0;
 d2=Pan[1].l/1000.0;
 d3=Pan[2].l/1000.0;
 p0=Pan[3].l;
 p2=Pan[4].l;
 x0=Pos[0].x;
 y0=Pos[0].y;
 x1=Pos[1].x;
 y1=Pos[1].y;
 x2=x1+p2+d2;
 y2=y0;
 x3=x0;
 y3=y1+p0+d0;
 //----cari perpotongan--------------------
 double pengali= (d0*(x2-x1) + d2*(y3-y1))/((y2-y1)*(x3-x1) - (y3-y1)*(x2-x1));
```

```
 X= x1 + (x3-x1)* pengali;
 Y= y1 + (y3-y1)* pengali;
 //-----cari posisi akhir pahatan--------------------
 int count=  (int)(y0-y3)/d3;
 if(count<1) count=1;
 double length= (double)(y0-y3)/count;
 char buffer[80];
 for(int i=1; i<=count; i++)
 {
   if((length*i) <= (y0-Y))
   {
     ya= (y0-y1) - (length*i);
     xa= x1 + ya*(x2-x1)/(y2-y1) + d2;
     ya= y1 + ya;
  //---gerak 1
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",ya, x0, F);
     mDest->Lines->Add(AnsiString(buffer));
  //---gerak 2a
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",ya, xa, F);
     mDest->Lines->Add(AnsiString(buffer));
   }
   else
   {
     ya= (length*i) - (y0-y1);
     xa= x1 + (ya-d0)*(x3-x1)/(y3-y1);
     ya= y1 - ya;
   //---gerak 1
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",ya, x0, F);
     mDest->Lines->Add(AnsiString(buffer));
   //---gerak 2a
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",ya, xa, F);
     mDest->Lines->Add(AnsiString(buffer));
   //---gerak 2b
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",Y, X, F);
     mDest->Lines->Add(AnsiString(buffer));
   }
  //---gerak 3
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",y2, x2, F);
     mDest->Lines->Add(AnsiString(buffer));
  //---gerak 4
     sprintf(buffer,"G01  X%.3f  Z%.3f  F%d ",y0, x0, F);
     mDest->Lines->Add(AnsiString(buffer));
 }
}
```

3.2. Penggabungan Unit Animasi dengan *User Interface*

Pada penggabungan unit ini menjelaskan proses eksekusi secara keseluruhan dari data *input* di *user interface* sampai unit animasi. Penggabungan yang dimaksud adalah menghubungkan semua modul pada setiap unit yang terdapat pada *form* utama. Sebagaimana yang telah dibahas di bab sebelumnya, bahwa tiap *form* yang

terdapat sub-rutin yang berisi modul-modul perintah. Penggabungan ini dilakukan dengan jalan menghubungkan perintah pemrograman dalam tiap modul. Setelah proses penggabungan selesai, maka semua perintah dari hasil data *input* di *user interface unit* dapat diterjemahkan oleh unit animasi menjadi proses permesinan CNC dalam bentuk simulasi, sehingga menghasilkan produk benda kerja.



Gambar 3.38. *Flow chart* eksekusi

Diagram alir pada gambar 3.38, menjelaskan bahwa eksekusi dari *list* program G codes hasil *input* dari *user interface* pertama dibaca parameter yang ada dalam setiap blok program. Setelah itu mengeksekusi tiap kode yang ada. Pada *software* ini untuk kode G dalam satu blok program hanya dapat diterjemahkan satu

kode, karena setiap baris program hanya dapat dikenali satu kode G saja dengan nilai parameternya. Penterjemahan terus berlangsung sampai perintah penutup program dibaca dan diterjemahkan (M30).