Y-12

Y/ME-94-08

OAK RIDGE Y-12 PLANT

MARTIN MARIETTA

Kearney & Trecker Milwaukee Matic 600

J. E. Seat
K. S. Rogers

Y-12 Equipment Design Dept.
Engineering Division

10/21/94

Prepared by the
Oak Ridge Y-12 Plant
Oak Ridge, Tennessee 37831
managed by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY
UCN-13872 (2 10-90)

MASTER

MAR 31 1995

OSTI

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

LIST OF FILES

(YME-94-08)

total 7

drwxrwxrwx	6 kru	user	512 Oct 7 09:18 .
drwxr-xr-x	23 kru	user	1024 Oct 7 09:39 ..
drwxr-xr-x	3 kru	user	512 Sep 20 14:06 MACHINES
drwxr-xr-x	3 kru	user	512 Sep 20 14:05 MIMIC
drwxr-xr-x	3 kru	user	512 Sep 20 13:55 PARTS
drwxr-xr-x	2 kru	user	512 Oct 7 10:03 SNAPSHOTS

KT_MM600/MACHINES:

total 3

drwxr-xr-x	3 kru	user	512 Sep 20 14:06 .
drwxrwxrwx	6 kru	user	512 Oct 7 09:18 ..
drwxr-xr-x	2 kru	user	512 Oct 3 10:10 KEARNEY_TRECKER

KT_MM600/MACHINES/KEARNEY_TRECKER:

total 13

drwxr-xr-x	2 kru	user	512 Oct 3 10:10 .
drwxr-xr-x	3 kru	user	512 Sep 20 14:06 ..
-rw-r--r--	1 kru	user	5328 Oct 3 10:10 kt_mm600_test

KT_MM600/MIMIC:

total 3

drwxr-xr-x	3 kru	user	512 Sep 20 14:05 .
drwxrwxrwx	6 kru	user	512 Oct 7 09:18 ..
drwxr-xr-x	2 kru	user	512 Sep 20 14:05 KEARNEY_TRECKER

KT_MM600/MIMIC/KEARNEY_TRECKER:

total 65

drwxr-xr-x	2 kru	user	512 Sep 20 14:05 .
drwxr-xr-x	3 kru	user	512 Sep 20 14:05 ..
-rwxr-xr-x	1 kru	user	31809 Sep 20 14:05 kt_mm600.mmc

KT_MM600/PARTS:

total 3

drwxr-xr-x	3 kru	user	512 Sep 20 13:55 .
drwxrwxrwx	6 kru	user	512 Oct 7 09:18 ..
drwxr-xr-x	4 kru	user	512 Sep 20 13:56 MM600

KT_MM600/PARTS/MM600:

total 4

drwxr-xr-x	4 kru	user	512 Sep 20 13:56 .
drwxr-xr-x	3 kru	user	512 Sep 20 13:55 ..
drwxr-xr-x	2 kru	user	512 Sep 20 13:55 HORZ_BED
drwxr-xr-x	2 kru	user	512 Sep 20 13:55 UPRIGHT

KT_MM600/PARTS/MM600/HORZ_BED:

total 60

drwxr-xr-x	2 kru	user	512 Sep 20 13:55 .
drwxr-xr-x	4 kru	user	512 Sep 20 13:56 ..
-rw-r--r--	1 kru	user	3236 Sep 20 13:55 base
-rw-r--r--	1 kru	user	6950 Sep 20 13:55 face_plate_2ta
-rw-r--r--	1 kru	user	7916 Sep 20 13:55 rotat_table
-rw-r--r--	1 kru	user	6082 Sep 20 13:55 second_table
-rw-r--r--	1 kru	user	2188 Sep 20 13:55 support_table2
-rw-r--r--	1 kru	user	809 Sep 20 13:55 table
-rw-r--r--	1 kru	user	466 Sep 20 13:55 table_base
-rw-r--r--	1 kru	user	447 Sep 20 13:55 upper_suppor_2

KT_MM600/PARTS/MM600/UPRIGHT:

total 53

drwxr-xr-x	2 kru	user	512 Sep 20 13:55 .
drwxr-xr-x	4 kru	user	512 Sep 20 13:56 ..
-rw-r--r--	1 kru	user	5919 Sep 20 13:55 base_assy
-rw-r--r--	1 kru	user	2561 Sep 20 13:55 floor_bed
-rw-r--r--	1 kru	user	759 Sep 20 13:55 kt_text
-rw-r--r--	1 kru	user	2178 Sep 20 13:55 spindle

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *www*

MASTER

-rw-r--r--	1 kru	user	1984 Sep 20 13:55	spindle_cap
-rw-r--r--	1 kru	user	2441 Sep 20 13:55	spindle_nose
-rw-r--r--	1 kru	user	8131 Sep 20 13:55	up_bed
-rw-r--r--	1 kru	user	305 Sep 20 13:55	y_cvr_scale

KT_MM600/SNAPSHOTS:

total 8607

drwxr-xr-x	2 kru	user	512 Oct 7 10:03	.
drwxrwxrwx	6 kru	user	512 Oct 7 09:18	..
-rw-r--r--	1 kru	user	602334 Oct 7 09:39	base_assy_w_supp
-rw-r--r--	1 kru	user	1892054 Oct 7 09:24	ktmm600_mach
-rw-r--r--	1 kru	user	1715951 Oct 7 09:26	ktmm600_mach2
-rw-r--r--	1 kru	user	194147 Oct 7 09:32	supp_tab2

NOTES

----- : Sequencing points for the controller same as N's
 ----- H's for operator use and is not defined in MIMIC file
 ----- E's for deep hole drilling and is not defined at this time
 ----- G88 is not defined & G83 & G87 needs some modification

(YME-94-08)

VAR

```

Next_tool_no      : INTEGER
Std_motion_added  : INTEGER
cli_str           : STRING
last_tool_name    : STRING
cli_real          : REAL
tmp_value         : REAL
code_letter       : STRING
current_position  : REAL
tmpreal           : REAL

```

CONST

```

ZERO_TOL          0.001
X_AXS             1
Y_AXS             2
Z_AXS             3
B_AXS             4
C_AXS             5
DELAY_TIME        2.0
stepsize          .3

```

STATE

```

g_Flood_coolant
g_mist_coolant

```

REGISTER

```

R_AXIS1,          UNIT_MULTIPLIER = TO_MM    --- X
R_AXIS2,          UNIT_MULTIPLIER = TO_MM    --- Y
R_AXIS3,          UNIT_MULTIPLIER = TO_MM    --- Z
R_AXIS4,          UNIT_MULTIPLIER = TO_DEG   --- A
R_AXIS5,          UNIT_MULTIPLIER = TO_DEG   --- C

R_FEEDRATE,       DEFAULT_VALUE = 100.0, UNIT_MULTIPLIER = TO_MM_PER_SEC,   MODAL
R_RAPIDRATE,      DEFAULT_VALUE = 239.6067, UNIT_MULTIPLIER = TO_MM_PER_SEC, MODAL
R_ROTATE,         DEFAULT_VALUE = 15.0,  UNIT_MULTIPLIER = TO_DEG_PER_SEC,   MODAL
R_MAX_CHORD_DEV,  DEFAULT_VALUE = 4

R_TO_MM,          DEFAULT_VALUE = 1,          MODAL
R_TO_DEG,         DEFAULT_VALUE = 1,          MODAL
R_TO_MM_PER_SEC,  DEFAULT_VALUE = 1,          MODAL
R_TO_DEG_PER_SEC, DEFAULT_VALUE = 1,          MODAL
R_TO_SEC,         DEFAULT_VALUE = 1.0,        MODAL
r_Clearance_plane, DEFAULT_VALUE = .100, UNIT_MULTIPLIER = TO_MM,   MODAL
H_CODE,          DEFAULT_VALUE = 1,          MODAL

```

OFFSET

```

o_fixture
o_origin_ofs
o_presets

```

CONTROL_CHARS

```

TYPE_II_START_CHARS: '('
TYPE_II_END_CHARS: ')'
IGNORE_CHARS: '\r\t %^'
END_OF_BLOCK_CHAR: '\n'
EOL_COMMENT_CHARS: '/'
IGNORE_BLOCK_END_CHAR: '\\
--COMMENT_START_CHARS: ':'
--COMMENT_END_CHARS: '\n'

```

FORWARD

```
canned_cycle_on_block :    ROUTINE : BOOLEAN
Unary_pos_rotation :      ROUTINE : REAL
Is_code_negative :        ROUTINE : BOOLEAN
```

CODE_DEFINITION

X, EXECUTE

BEGIN

```
SET_REGISTER( R_AXIS1, CODE_VALUE( 'X' ) )
Standard_motion()
```

END

Y, EXECUTE

BEGIN

```
SET_REGISTER( R_AXIS2, CODE_VALUE( 'Y' ) )
Standard_motion()
```

END

Z, EXECUTE

BEGIN

```
SET_REGISTER( R_AXIS3, CODE_VALUE( 'Z' ) )
Standard_motion()
```

END

B, EXECUTE

BEGIN

```
tmp_value = Unary_pos_rotation( REF_POSITION( B_AXIS ),CODE_VALUE('B'), 'B' )
-- WRITE("tmp_value = ",tmp_value,cr) -- ksr added
SET_REGISTER( R_AXIS4, tmp_value )
-- WRITE("R_AXIS4 = ",REGISTER_VALUE(R_AXIS4),cr,cr) --ksr added
Standard_motion()
```

END

C, EXECUTE

BEGIN

```
tmp_value = Unary_pos_rotation( REF_POSITION( C_AXIS ),CODE_VALUE('C'), 'C' )
SET_REGISTER( R_AXIS5, tmp_value )
Standard_motion()
```

END

I, EXECUTE -- arc center offset

BEGIN

```
SET_REGISTER( R_X_DISTANCE, CODE_VALUE( 'I' ) )
```

END

J, EXECUTE -- arc center offset

BEGIN

```
SET_REGISTER( R_Y_DISTANCE, CODE_VALUE( 'J' ) )
```

END

K, EXECUTE -- arc center offset

BEGIN

```
SET_REGISTER( R_Z_DISTANCE, CODE_VALUE( 'K' ) )
```

END

L, EXECUTE

BEGIN

```
SET_REGISTER( R_LENGTH, CODE_VALUE( 'L' ) )
```

END

F, EXECUTE -- dwell or feedrate

BEGIN

```
IF( IS_CODE_ON_BLOCK( 'G', 04 ) ) THEN
SET_REGISTER( R_ELAPSED_TIME, CODE_VALUE( 'F' ) )
ELSE
```

```

        SET_REGISTER( R_FEEDRATE, CODE_VALUE( 'F' ) )
    ENDIF
END

S, EXECUTE -- spindle speed
BEGIN
    SET_REGISTER( R_SPINRATE, CODE_VALUE( 'S' ) )
END

T, EXECUTE -- toolselect
BEGIN
    Next_tool_no = CODE_VALUE( 'T' )
END

R, EXECUTE -- clearance plane in canned cycle or radius for circular interpolation
BEGIN
    IF( canned_cycle_on_block( ) ) THEN
        SET_REGISTER( r_Clearance_plane, CODE_VALUE( 'R' ) )
    ELSE
        SET_REGISTER( R_RADIUS, CODE_VALUE( 'R' ) )
    ENDIF
END

N, EXECUTE
BEGIN
    SET_REGISTER( R_BLOCK_NUMBER, CODE_VALUE( 'N' ) )
END

O, EXECUTE
BEGIN
    SET_REGISTER( R_BLOCK_NUMBER, CODE_VALUE( 'O' ) )
END

D, EXECUTE
BEGIN
    SET_REGISTER( R_CUTTER_RADIUS_COMP, CODE_VALUE( 'D' ) )
END

E, EXECUTE -- fixed cycle incremental feed distance (not defined)
BEGIN
END

G00, EXECUTE
BEGIN
    SET_STATE( G_MOTION_TYPE, S_RAPID )
    SET_STATE( G_SLEW, S_ON )
    SET_STATE( G_STOCK_UPDATE, S_OFF )
END

G01, EXECUTE
BEGIN
    SET_STATE( G_MOTION_TYPE, S_FEED )
    SET_STATE( G_SLEW, S_OFF )
    SET_STATE( G_STOCK_UPDATE, S_ON )
END

G02, EXECUTE
BEGIN
    SET_STATE( G_MOTION_TYPE, S_CW2 )
    SET_STATE( G_SLEW, S_OFF )
    SET_STATE( G_STOCK_UPDATE, S_ON )
END

G03, EXECUTE
BEGIN
    SET_STATE( G_MOTION_TYPE, S_CCW2 )

```



```

        SET_STATE( G_SLEW, S_OFF )
        SET_STATE( G_STOCK_UPDATE, S_ON )
    END

G04, EXECUTE
    BEGIN
        DWELL()
    END

G17, EXECUTE
    BEGIN
        SET_STATE( G_PLANE, S_X_Y )
    END

G18, EXECUTE
    BEGIN
        SET_STATE( G_PLANE, S_X_Z )
    END

G19, EXECUTE
    BEGIN
        SET_STATE( G_PLANE, S_Y_Z )
    END

G40, EXECUTE
    BEGIN
        SET_STATE( G_CUTTER_COMP, S_OFF )
    END

G41, EXECUTE
    BEGIN
        SET_STATE( G_CUTTER_COMP, S_LEFT )
    END

G42, EXECUTE
    BEGIN
        SET_STATE( G_CUTTER_COMP, S_RIGHT )
    END

G48, EXECUTE --fixture offset
    BEGIN
-- not defined
    END

G70, EXECUTE --Inch mode
    BEGIN
        SET_REGISTER( R_TO_MM, 25.4 ) --- working in inches
        SET_REGISTER( R_TO_MM_PER_SEC, 25.4/60.0 ) --- working in in/min
    END

G71, EXECUTE --Metric mode
    BEGIN
        SET_REGISTER( R_TO_MM, 1 ) --- working in mm
        SET_REGISTER( R_TO_MM_PER_SEC, 1.0/60.0 ) --- working in mm/min
    END

G80, EXECUTE --- fixed cycle; cancel
    BEGIN
        cs_cancel()
    END

G81, EXECUTE --- fixed cycle; drill
    BEGIN
        cs_drill()
    END

```

```

G82, EXECUTE    --- fixed cycle; counterbore or spotfacing
  BEGIN
    cs_spotface()
  END

G83, EXECUTE    --- fixed cycle; chip breaking
  BEGIN
    cs_chip_break()
  END

G84, EXECUTE    --- fixed cycle; tapping
  BEGIN
    cs_tap()
  END

G85, EXECUTE    --- fixed cycle; reaming
  BEGIN
    cs_ream()
  END

G86, EXECUTE    --- fixed cycle; bore in, stop spindle, rapid out
  BEGIN
    cs_bore()
  END

G87, EXECUTE    --- fixed cycle; deep hole drilling
  BEGIN
    cs_deep_hole_drill()
  END

G89, EXECUTE    --- fixed cycle; bore in, dwell, bore out
  BEGIN
    cs_bore_dwell()
  END

G90, EXECUTE
  BEGIN
    SET_STATE( G_POSITION, S_ABSOLUTE )
  END

G91, EXECUTE
  BEGIN
    SET_STATE( G_POSITION, S_RELATIVE )
  END

G93, EXECUTE
  BEGIN
--    SET_STATE( G_SPINDLE_FEED_MODE, S_INVERSE_TIME )
  END

G94, EXECUTE
  BEGIN
--    SET_STATE( G_SPINDLE_FEED_MODE, S_ON )
  END

G95, EXECUTE
  BEGIN
    SET_STATE( G_SPINDLE_FEED_MODE, S_ON )
  END

G98, EXECUTE
  BEGIN
    if (IS_CODE_ON_BLOCK('X')) then
      tmpreal = (REGISTER_VALUE(R_AXIS1)/25.4)-REF_POSITION(1)
      --write ("tmpreal = ",tmpreal,cr)
    end if
  END

```

```

        Change_an_offset (o_presets, 1,-tmpreal )
    endif

    if (IS_CODE_ON_BLOCK('Y')) then
        tmpreal = (REGISTER_VALUE(R_AXIS2)/25.4)-REF_POSITION(2)
        --write ("tmpreal = ",tmpreal,cr)
        Change_an_offset (o_presets, 2,-tmpreal )
    endif

    if (IS_CODE_ON_BLOCK('Z')) then
        tmpreal = (REGISTER_VALUE(R_AXIS3)/25.4)-REF_POSITION(3)
        --write ("tmpreal = ",tmpreal,cr)
        Change_an_offset (o_presets, 3,-tmpreal )
    endif

    ENABLE_OFFSET( o_presets )
END

G99, EXECUTE
BEGIN
    DISABLE_OFFSET(o_presets)
END

M00, EXECUTE -- program stop
BEGIN
    CYCLE_STOP()
END

M01, EXECUTE -- optional program stop
BEGIN
    IF( IS_STATE_SET( G_OPT_STOP, S_ON ) ) THEN
        CYCLE_STOP()
    ENDIF
END

M02, EXECUTE -- end of program
BEGIN
    SET_STATE( G_STOCK_UPDATE, S_OFF )

    DISABLE_OFFSET(o_presets)
    POS_AXIS_NOSIMUL( X_AXS, 0 )
    POS_AXIS_NOSIMUL( Y_AXS, 0 )
    POS_AXIS_NOSIMUL( Z_AXS, 0 )
    POS_AXIS_NOSIMUL( B_AXS, 0 )
    POS_AXIS_NOSIMUL( C_AXS, 0 )

    SET_STATE( G_SPINDLE_MODE, S_STOPPED )
    SET_STATE( g_Mist_coolant, S_OFF )
    SET_STATE( g_Flood_coolant, S_OFF )

    last_tool_name = STR("T%g", REGISTER_VALUE (R_TOOL_NO) )
    ATTACH_DEVICE_AT_FRAME (last_tool_name, 'toolchanger')

    CYCLE_STOP()
END

M03, EXECUTE -- spindle clockwise
BEGIN
    SET_STATE( G_SPINDLE_MODE, S_CW )
END

M04, EXECUTE -- spindle counterclockwise
BEGIN
    SET_STATE( G_SPINDLE_MODE, S_CCW )
END

```

```

M05, EXECUTE -- spindle stop
BEGIN
    SET_STATE( G_SPINDLE_MODE, S_STOPPED )

    SET_STATE( g_Mist_coolant, S_OFF )
    SET_STATE( g_Flood_coolant, S_OFF )
END

M06, EXECUTE -- tool transfer
BEGIN
    Load_tool( )
END

M07, EXECUTE -- mist coolant on
BEGIN
    SET_STATE( g_Mist_coolant, S_ON )
END

M08, EXECUTE -- flood coolant on
BEGIN
    SET_STATE( g_Flood_coolant, S_ON )
END

M09, EXECUTE -- coolant off
BEGIN
    SET_STATE( g_Mist_coolant, S_OFF )
    SET_STATE( g_Flood_coolant, S_OFF )
END

M19, EXECUTE -- keylock
BEGIN
-- not defined
END

M25, EXECUTE -- bypass axis offsets
BEGIN
-- not defined
END

M30, EXECUTE -- end of program
BEGIN
    SET_STATE( G_STOCK_UPDATE, S_OFF )

    DISABLE_OFFSET(o_presets)

    SET_STATE( G_SPINDLE_MODE, S_STOPPED )
    SET_STATE( g_Mist_coolant, S_OFF )
    SET_STATE( g_Flood_coolant, S_OFF )

last_tool_name = STR("T%g", REGISTER_VALUE (R_TOOL_NO) )
ATTACH_DEVICE_AT_FRAME (last_tool_name, 'toolchanger')

    CYCLE_STOP()
END

----- CUSTOM CODES -----

H, EXECUTE
BEGIN
    SET_REGISTER( H_CODE, 10)
END

--:, EXECUTE
--BEGIN
--SET_REGISTER( R_BLOCK_NUMBER, CODE_VALUE( 'N' ) )
--END

```

```

-----
-----BEGIN APPLICATION ESCAPE PROCEDURES -----
-----BEGIN APPLICATION ESCAPE PROCEDURES -----
-----BEGIN APPLICATION ESCAPE PROCEDURES -----

```

```

PROCEDURE
INITIALIZE_CYCLE( )

```

```

-----
BEGIN

```

```

    SET_STATE( G_CUTTING_MODE, S_MILLING )
    SET_STATE( G_MOTION_TYPE, S_FEED )
    SET_STATE( G_SLEW, S_OFF )
    SET_STATE( G_EXACT_STOP, S_ON )
    OPEN WINDOW 'OPERATOR MESSAGES' @ .5,.5: 1 AS 1

```

```

    CLI( "TRACE DELETE_ALL" )

```

```

    Std_motion_added      = FALSE
    Next_tool_no          = 0

```

```

    -- these lines make the part update when the program starts

```

```

    SET_STATE( G_STOCK_UPDATE, S_ON )
    SET_STATE( G_STOCK_UPDATE, S_OFF )
    SET_TOOL( TOOL_NAME() )
    SET_WORKPIECE( WORKPIECE_NAME() )

```

```

-- ( "WP=", WORKPIECE_NAME(), CR )

```

```

END INITIALIZE_CYCLE

```

```

-----
PROCEDURE
FINALIZE_CYCLE()

```

```

-----
BEGIN

```

```

    WRITE( "Finalizing program", CR )

```

```

END FINALIZE_CYCLE

```

```

-----
PROCEDURE

```

```

EOB()

```

```

VAR

```

```

-----
--This Procedure is called after all code processing for the block is complete
--but prior to the block's animation.

```

```

-----
BEGIN

```

```

    Std_motion_added      = FALSE
    UPDATE_GRAPHICS()

```

```

END EOB

```

```

-----BEGIN LOCAL PROCEDURES -----
-----BEGIN LOCAL PROCEDURES -----
-----BEGIN LOCAL PROCEDURES -----

```

```

PROCEDURE
Standard_motion()

```

```

-----
BEGIN

```

```

    IF( Std_motion_added == TRUE OR canned_cycle_on_block( ) ) THEN

```

```

    RETURN
ENDIF

STD_MOTION( )

Std_motion_added = TRUE
END Standard_motion
-----

PROCEDURE
Load_tool()
VAR
last_tool_name , next_tool_name :STRING
tl: REAL
-----
BEGIN
    SET_STATE( G_STOCK_UPDATE, S_OFF )
    SAVE_STATE( G_MOTION_TYPE)
    SET_STATE( G_MOTION_TYPE, S_RAPID)

-- WORKPIECE_CLEANUP()           -- commented this out because all
-- SET_TOOL (Tool_name())         -- previous material removal disapears
-- SET_WORKPIECE(WORKPIECE_NAME()) -- at each tool change
-- SET_STATE( G_STOCK_UPDATE, S_ON)

last_tool_name = STR("T%g", REGISTER_VALUE (R_TOOL_NO) )
next_tool_name = STR("T%g", Next_tool_no)

ATTACH_DEVICE_AT_FRAME (last_tool_name, 'toolchanger')
ATTACH_DEVICE_AT_FRAME (next_tool_name, 'gauge_point')
CLI("TRANSLATE '" + next_tool_name + "' BY 0, 0, " + str('%g',TOOL_LENGTH(next_tool_r

SET_TOOL (next_tool_name)

SET_REGISTER (R_TOOL_NO, Next_tool_no)

RESTORE_STATE( G_MOTION_TYPE )
SET_STATE( G_STOCK_UPDATE, S_ON )

END Load_tool
-----

PROCEDURE
TYPE_II( type_ii_string      : STRING )
VAR
Keyword, command_str        :STRING
x_ofs, y_ofs, z_ofs         :REAL
start_num, end_num          :INTEGER
-----
BEGIN
    keyword = LEFTSTR( type_ii_string, 3)
    command_str = RIGHTSTR( type_ii_string, (LEN( type_ii_string ) - 4) )

    SWITCH(keyword )

    case 'ACP': -- Specify and Modify Adaptive Control
        continue_case
    case 'acp':

    case 'CER': -- Continue part program Execution Upon Device Errors
        continue_case
    case 'cer':

    case 'CLF': -- Close File

```

```

        continue_case
case 'clf':

case 'CLS':  -- Call Subroutine
    continue_case
case 'cls':

case 'CPY':  -- Copy A File
    continue_case
case 'cpy':

case 'DAT':  -- Update Date and Time
    continue_case
case 'dat':

case 'DFC':  -- Define a Cycle
    continue_case
case 'dfc':

case 'DFS':  -- Define a Subroutine
    continue_case
case 'dfs':

case 'DLF':  -- Delete a File
    continue_case
case 'dlf':

case 'DLP':  -- Delete a Parameter
    continue_case
case 'dlp':

case 'END':  -- End a Statement
    continue_case
case 'end':

case 'EVT':  -- Record Part Program Event
    continue_case
case 'evt':

case 'FWR':  -- Write to File
    continue_case
case 'fwr':

case 'FXC':  -- Load to Fixture Offset Table
    continue_case
case 'fxc':

case 'GTO':  -- Go to Label
    continue_case
case 'gto':

case 'IFT':  -- Conditional "If True"
    continue_case
case 'ift':

case 'LBL':  -- Label Definition
    continue_case
case 'lbl':

case 'MIR':  -- Mirror Image, Axis Inversion
    continue_case
case 'mir':

case 'NAM':  -- Name a Part Program
    continue_case
case 'nam':

```

```

case 'OIF':    -- Name a Spindle Probe Inspection File
    continue_case
case 'oif':

case 'OPF':    -- Open a File
    continue_case
case 'opf':

case 'PAR':    -- Parameter Assignment Statement
    continue_case
case 'par':

case 'PRT':    -- Print File
    continue_case
case 'prt':

case 'RCP':    -- Copy a File to or from a Remote Station
    continue_case
case 'rcp':

case 'SER':    -- Stop Program Execution Upon Device Errors
    continue_case
case 'ser':

case 'SET':    -- Set an Output
    continue_case
case 'set':

case 'TLD':    -- Load a Tool Table with Tool Parameters
    continue_case
case 'tld':

case 'U88':    -- Programmable Boring Orientation Cycle
    continue_case
case 'u88':

case 'UAT':    -- Select Alternate Tool Data
    continue_case
case 'uat':

case 'UCY':    -- Maximum Part Program Execution Time - FMS
    continue_case
case 'ucy':

case 'UFX':    -- Control Calculated Fixture Offset
    continue_case
case 'ufx':

case 'ULF':    -- Recalls Saved Fixture Offset Table
    continue_case
case 'ulf':

case 'USF':    -- Fixture Offset Table Saved on Mass Storage Media
    continue_case
case 'usf':

case 'UTL':    -- Identifies All Tools Used in a Part Program - FMS
    continue_case
case 'utl':

case 'WAI':    -- Wait on Input
    continue_case
case 'wai':

case 'ZTB':    -- Zero Either Tool Table or Fixture Offset Table or Both

```



```

    continue_case
case 'ztb':

case 'MSG': -- Programmable Operator Message
    continue_case
case 'msg':
    WRITE(@1, command_str, CR )
case 'FXL': -- Fixture offset, Entered by operator
    continue_case
case 'fxl' :
    --Set the current offsets
    x_ofs = OFFSET_VALUE(o_fixture, 1 )
    y_ofs = OFFSET_VALUE(o_fixture, 2 )
    z_ofs = OFFSET_VALUE(o_fixture, 3 )

    READ_KBD( "ENTER fixture offsets (x y z )", x_ofs, y_ofs, z_ofs )
    CHANGE_AN_OFFSET(o_fixture, 1, x_ofs )
    CHANGE_AN_OFFSET(o_fixture, 2, y_ofs )
    CHANGE_AN_OFFSET(o_fixture, 3, z_ofs )

default:
    WRITE( "Invalid Type II Keyword <", keyword, ">", CR )
ENDSWITCH
END TYPE_II
-----

PROCEDURE
cs_cancel() --G80
-----

BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
        case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
            POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
    ENDSWITCH

    POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

    RESTORE_STATE( G_MOTION_TYPE )
END
-----

PROCEDURE --G81
cs_drill()
-----

BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
        case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))

```

```

    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
        POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
ENDSWITCH

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

SET_STATE( G_MOTION_TYPE, S_RAPID )

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

RESTORE_STATE( G_MOTION_TYPE )
END
-----

PROCEDURE
cs_spotface() --G82
-----
BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
        case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
            POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
    ENDSWITCH

    POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

    SET_STATE( G_MOTION_TYPE, S_FEED )
    SET_STATE( G_SLEW, S_OFF )
    SET_STATE( G_STOCK_UPDATE, S_ON )

    INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

    DELAY( DELAY_TIME )

    SET_STATE( G_MOTION_TYPE, S_RAPID )

    POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

    RESTORE_STATE( G_MOTION_TYPE )
END
-----
-- this is not exactly how G83 works, modify later to use 'E' words --

PROCEDURE
cs_chip_break() --G83
-----
BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_FEED )

    SWITCH TRUE -- if we need to move x or y do it
        case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
            POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))

```

```

    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
    POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
ENDSWITCH

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

DELAY( DELAY_TIME )

```

```

SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

RESTORE_STATE( G_MOTION_TYPE )
END
-----

```

```

PROCEDURE
cs_tap() --G84
-----

```

```

BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
        POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
        POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
    case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
        POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
        POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
    ENDSWITCH

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

RESTORE_STATE( G_MOTION_TYPE )
END
-----

```

```

PROCEDURE
cs_ream() --G85
-----

```

```

BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
        POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
        POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))

```

```

case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
  POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
  POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
ENDSWITCH

```

```

POS_AXIS_NOSIMUL( Y_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

RESTORE_STATE( G_MOTION_TYPE )
END

```

```

-----
PROCEDURE
cs_bore() --G86

```

```

-----
BEGIN

```

```

  SAVE_STATE( G_MOTION_TYPE )
  SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

  SWITCH TRUE -- if we need to move x or y do it
  case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
    POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
  case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
  case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
    POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
  ENDSWITCH

```

```

  POS_AXIS_NOSIMUL( Y_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

  SET_STATE( G_MOTION_TYPE, S_FEED )
  SET_STATE( G_SLEW, S_OFF )
  SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

  INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

  SET_STATE( G_SPINDLE_MODE, S_STOPPED )
  SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

  POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

  RESTORE_STATE( G_MOTION_TYPE )

```

```

END

```

```

-----
-- modify later to use 'E' words --

```

```

PROCEDURE --G87
cs_deep_hole_drill()

```

```

-----
BEGIN

```

```

  SAVE_STATE( G_MOTION_TYPE )
  SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

  SWITCH TRUE -- if we need to move x or y do it
  case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
    POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))

```

```

    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
    POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
    POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
ENDSWITCH

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

SET_STATE( G_MOTION_TYPE, S_RAPID )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

RESTORE_STATE( G_MOTION_TYPE )
END

```

```

-----
PROCEDURE
cs_bore_dwell() --G89

```

```

-----
BEGIN
    SAVE_STATE( G_MOTION_TYPE )
    SET_STATE( G_MOTION_TYPE, S_RAPID )

    SWITCH TRUE -- if we need to move x or y do it
    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == TRUE):
        POS_AXIS_SIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
        POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
    case (Is_code_on_block('X') == FALSE AND Is_code_on_block('Y') == TRUE):
        POS_AXIS_NOSIMUL(Y_AXIS, REGISTER_VALUE(R_AXIS2))
    case (Is_code_on_block('X') == TRUE AND Is_code_on_block('Y') == FALSE):
        POS_AXIS_NOSIMUL(X_AXIS, REGISTER_VALUE(R_AXIS1))
    ENDSWITCH

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

SET_STATE( G_MOTION_TYPE, S_FEED )
SET_STATE( G_SLEW, S_OFF )
SET_STATE( G_STOCK_UPDATE, S_ON )

```

```

INCR_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(R_AXIS3) )

```

```

DELAY( DELAY_TIME )

```

```

POS_AXIS_NOSIMUL( Z_AXIS, REGISTER_VALUE(r_Clearance_plane ) )

```

```

RESTORE_STATE( G_MOTION_TYPE )
END

```

```

ROUTINE
canned_cycle_on_block( ) : BOOLEAN
BEGIN
  IF( IS_CODE_ON_BLOCK( 'G', 80 ) OR IS_CODE_ON_BLOCK( 'G', 81 ) OR \
      IS_CODE_ON_BLOCK( 'G', 82 ) OR IS_CODE_ON_BLOCK( 'G', 83 ) OR \
      IS_CODE_ON_BLOCK( 'G', 84 ) OR IS_CODE_ON_BLOCK( 'G', 85 ) OR \
      IS_CODE_ON_BLOCK( 'G', 86 ) OR IS_CODE_ON_BLOCK( 'G', 87 ) OR \
      IS_CODE_ON_BLOCK( 'G', 89 ) OR IS_CODE_ON_BLOCK( 'G', 98 ) ) THEN
    RETURN( TRUE )
  ELSE
    RETURN( FALSE )
  ENDIF
END canned_cycle_on_block

```

```

-----
--
--                               Is_code_negative
--
-- This routine is required to check for a minus sign immediately after the
-- passed code letter. This is necessary because the CNC Code processing
-- mechanism returns a value of 0 or -0.
--
-----

```

```

ROUTINE
Is_code_negative( code_letter : STRING ) : BOOLEAN

VAR
  blk_string : STRING
  ii         : INTEGER

-----
BEGIN
  code_letter = TOUPPER( LEFTSTR( code_letter, 1 ) )

  blk_string = TOUPPER( BLOCK_STRING() )

  FOR ii = 1 to LEN( blk_string ) DO
    IF( SUBSTR( blk_string, ii, 1 ) == code_letter ) THEN
      IF( SUBSTR( blk_string, ii+1, 1 ) == '-' ) THEN
        RETURN( TRUE )
      ELSE
        RETURN( FALSE )
      ENDIF
    ENDIF
  ENDFOR

  -- The code letter is not on the current block
  RETURN ( FALSE )
END Is_code_negative

```

```

-----
--
--                               Unary Positional Rotation
--
-- This Mimic Routine will convert rotation direction denoted by the sign into
-- the standard absolute positioning mode supported by VNC when in absolute
-- mode.
--
-----

```

```

ROUTINE
Unary_pos_rotation( current_position, code_val:REAL; code_letter:STRING ) : REAL

```

```

VAR
tmp_value : REAL

BEGIN
  IF( IS_STATE_SET( G_POSITION, S_RELATIVE ) ) THEN
    RETURN( code_val )
  ENDIF

  code_letter = TOUPPER( LEFTSTR( code_letter, 1 ) )

  IF( (current_position MOD 360) == ABS( code_val ) ) THEN
    -- The B axis is already at the requestd location.
    -- Command it to move to the current position.
    tmp_value = current_position

  ELSE
    -- The sign denotes the direction of B axis rotation
    IF( code_val < 0.0 OR Is_code_negative( code_letter ) ) THEN
      -- The move is Clockwise

      --Make the negative value positive.
      tmp_value = code_val * -1.0

      -- Insure that tmp_value is less than the ref_position
      WHILE( current_position < tmp_value ) DO
        tmp_value = tmp_value - 360.0
      ENDWHILE

      -- Insure and greater than the (ref_position-360)
      WHILE( (current_position - 360.0) > tmp_value ) DO
        tmp_value = tmp_value + 360.0
      ENDWHILE
    ELSE
      -- The move is Counter Clockwise
      tmp_value = code_val

      -- Insure that tmp_value is greater than the ref_position
      WHILE( current_position > tmp_value ) DO
        tmp_value = tmp_value + 360.0
      ENDWHILE

      -- Insure that tmp_value is less than the (ref_position+360)
      WHILE( (current_position + 360.0) < tmp_value ) DO
        tmp_value = tmp_value - 360.0
      ENDWHILE
    ENDIF
  ENDIF

  RETURN( tmp_value )
END Unary_pos_rotation
-----

```


KT-MM600/SNAPSHOTS /ktmm600-mach

VMF-94-09

KT-MM600/SNAPSHOTS/ktmm600-mach2

KT-MMG00/SNAPSHOTS/base-assy-w-supp

YME-94-08

Message Window
Function aborted.

KT-MMG00/SNAPSHOTS/supp-tab2

YME-94-08

Distribution:

A. K. Lee/DOE-OSTI, 9731, MS-8175 (2)

K. S. Rogers, 9112, MS-8202

Y-12 Central Files, 9711-5, MS-8169