Bijlage 1: PLC Programma

```
0001 FUNCTION_BLOCK AxisControl
0002 (*-----------------------------------------------------------INPUTS-----------------------------------------------------------*)
0003 VAR_INPUT
0004     E_STOP: BOOL;
0005     lrManualSpeed: LREAL;              (*Defines manual movement speed of axis*)
0006     lrVelocity: LREAL;                     (*Defines maximum velocity of axis*)
0007     lrAcceleration: LREAL;                 (*Defines acceleration of axis*)
0008     lrDeceleration: LREAL;                 (*Defines Deceleration of axis*)
0009     lrJerk: LREAL;                     (*Defines Jerk of axis*)
0010     bCalibrateAll: BOOL;                   (*Impulse signal, used to callibrate all axis at once*)
0011     bAddToGroup: BOOL;                     (*Adds axis to group, so it can be used with NC programming*)
0012     bRemoveFromGroup: BOOL;                (*Removes axis from group, so it can be calibrated or used manually*)
0013     bActivateAllAxis: BOOL;                (*Activates all axis at once*)
0014     bAxisCouplingVis: BOOL;                (*Couples axis via user interface*)
0015     bCalibrationDistance: INT;             (*Distance to be traveled before calibration, if homingsensor is toggled*)
0016     bResetAllAxis: BOOL;                   (*Resets all axis*)
0017 END_VAR
0018 (*-----------------------------------------------------------OUTPUTS----------------------------------------------------------*)
0019 VAR_OUTPUT
0020     bSafeToPick: BOOL;                     (*Safe to pick up by robot at exact zero*)
0021 END_VAR
0022 (*-----------------------------------------------------------INTERNAL---------------------------------------------------------*)
0023 VAR
0024     EnableAxis: MC_Power;              (*FB to power up drive*)
0025     ReconfigGroup: CfgReconfigGroup;       (*FB to reconfigure group*)
0026     ManualMove: MC_Jog;                    (*FB to move the axis manually*)
0027     Calibrate: MC_Home;                    (*FB callibration*)
0028     Halt: MC_Halt;                     (*FB to stop all movement in a controlled way*)
0029     AddAxisToGroup: CfgAddAxisToGroup; (*FB to add axis to group*)
0030     MoveRelative: MC_MoveRelative;         (*FB to move the axis a certain distance relative to the current position*)
0031     MoveAbsolute: MC_MoveAbsolute;         (*FB to move the axis to an absolute location*)
0032     ResetAxis: MC_Reset;                   (*FB to reset drive after error*)
0033     SetPosition: MC_SetPosition;           (*FB Sets current position, used to bypass limit switches before calibration*)
0034
0035     bResetPosition: BOOL;                  (*Input for MC_SetPosition*)
0036     bCalibrationAllowed: BOOL;             (*TRUE if callibration can be carried out without moving beforehand*)
0037     bMoveBeforeCalibration: BOOL;          (*TRUE if calibration can only be carried out after moving away from sensor*)
0038     bCalibrateAfterMove: BOOL;         (*TRUE after move, used to start callibration once it is safe to do so*)
0039     bBusy: BOOL;                           (*Busy signal, not used*)
0040     lrActualManualSpeed: LREAL;            (*Variable to control the speed of manual movement*)
0041
0042     axisLinkExt AT %I*: axisLinkExtInputs;     (*STRUCT*)
0043     axisLinkInt: axisLinkIntInputs;            (*STRUCT*)
0044
0045     Axis: AXIS_REF;                        (*Coupling of NC axis*)
0046     activateAxisTON: TON;                  (*Time delay for activation of axis, safety reasons*)
0047     bAxisActive: BOOL;                 (*TRUE when axis is activated*)
0048     bCalibrate: BOOL;                      (*TRUE when axis is in need of calibration*)
0049     bCheckPosition: BOOL;                  (*Used to determine wether axis needs to move before calibration*)
0050     bAddAxisToGroup: BOOL;                 (*Adds axis to group*)
0051
0052     axisCouplingVisR_TRIG: R_TRIG;         (*TRIG to couple/decouple axis*)
0053 END_VAR
0054 VAR_IN_OUT
0055 END_VAR
```

```
0001 activateAxisTON(IN:= axisLinkInt.bEnableAxis OR bActivateAllAxis, PT:= t#1s, Q=> bAxisActive, ET=> );
0002
0003 IF bAxisActive AND NOT E_STOP AND NOT axisLinkExt.bCalibrated THEN
0004     bCalibrate := TRUE;
0005 END_IF
0006
0007 IF bCalibrate AND NOT E_STOP  THEN
0008     bResetPosition := TRUE;
0009 END_IF
0010
0011 SetPosition(
0012     Execute:= bResetPosition,
```

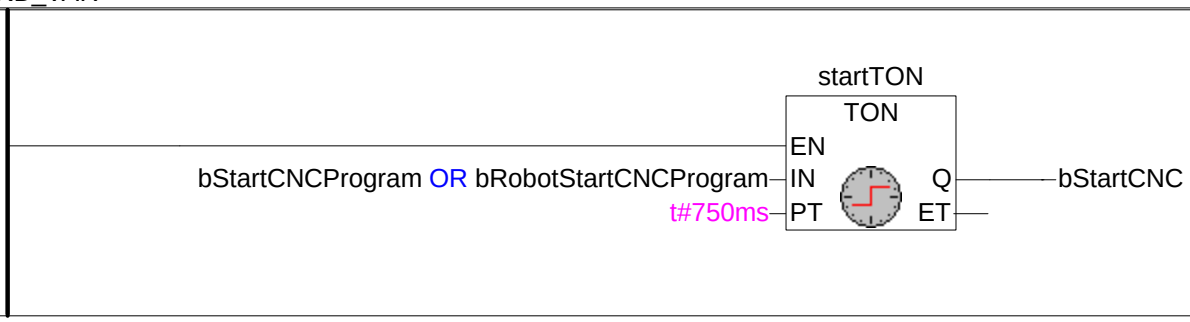
```
0013        Position:= 0,
0014        Mode:= ,
0015        Options:= ,
0016        Axis:= Axis,
0017        Done=> bCheckPosition,
0018        Busy=> ,
0019        Error=> ,
0020        ErrorID=> );
0021
0022 IF bCheckPosition AND NOT axisLinkExt.bLimitSwitch AND NOT E_STOP THEN
0023        bMoveBeforeCalibration := TRUE;
0024 END_IF
0025
0026 MoveRelative(
0027        Execute:= bMoveBeforeCalibration,
0028        Distance:= bCalibrationDistance,
0029        Velocity:= lrVelocity,
0030        Acceleration:= lrAcceleration,
0031        Deceleration:= lrDeceleration,
0032        Jerk:= lrJerk,
0033        BufferMode:= ,
0034        Options:= ,
0035        Axis:= Axis,
0036        Done=> ,
0037        Busy=> bBusy,
0038        Active=> ,
0039        CommandAborted=> ,
0040        Error=> ,
0041        ErrorID=> );
0042
0043 IF bCheckPosition AND axisLinkExt.bLimitSwitch AND NOT E_STOP AND NOT bBusy THEN
0044        bCalibrationAllowed := TRUE;
0045 END_IF
0046
0047 Calibrate(
0048        Execute:= NOT E_STOP AND bCalibrationAllowed,
0049        Position:= 0,
0050        HomingMode:= MC_DefaultHoming,
0051        bCalibrationCam:= axisLinkExt.bLimitSwitch,
0052        Axis:= Axis,
0053        Done=> bResetAllAxis,
0054        Busy=> ,
0055        Active=> ,
0056        CommandAborted=> ,
0057        Error=> ,
0058        ErrorID=> );
0059
0060 IF axisLinkExt.bCalibrated OR NOT (axisLinkInt.bEnableAxis OR bActivateAllAxis) THEN
0061        bCalibrate := FALSE;
0062        bResetPosition := FALSE;
0063        bCalibrationAllowed := FALSE;
0064        bMoveBeforeCalibration := FALSE;
0065        bCalibrateAfterMove := FALSE;
0066 END_IF
0067
0068 IF Axis.NcToPlc.ActPos < 1 AND axisLinkExt.bCalibrated THEN
0069        bSafeToPick := TRUE;
0070 ELSE
0071        bSafeToPick := FALSE;
0072 END_IF
0073
0074 IF axisLinkExt.bRapid THEN
0075        lrActualManualSpeed := 9.5;
0076 ELSE
0077        lrActualManualSpeed := lrManualSpeed;
0078 END_IF
0079
```

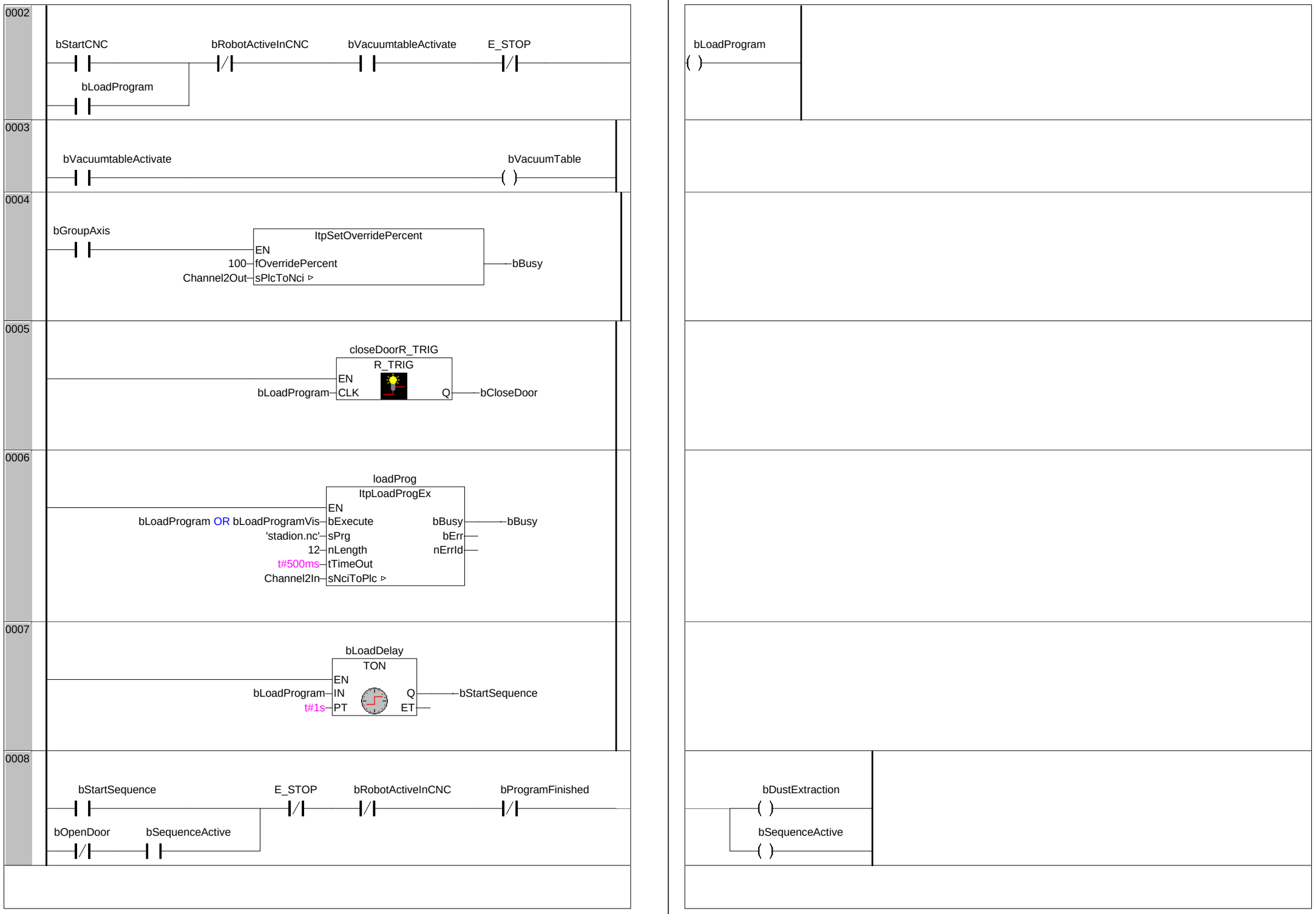
```
0080 EnableAxis(
0081     Enable:= (axisLinkInt.bEnableAxis OR bActivateAllAxis) AND NOT E_STOP,
0082     Enable_Positive:= (axisLinkInt.bEnableAxis OR bActivateAllAxis) AND NOT E_STOP,
0083     Enable_Negative:= (axisLinkInt.bEnableAxis OR bActivateAllAxis) AND NOT E_STOP,
0084     Override:= ,
0085     BufferMode:= ,
0086     Axis:= Axis,
0087     Status=> ,
0088     Busy=> ,
0089     Active=> ,
0090     Error=> ,
0091     ErrorID=> );
0092
0093 ResetAxis(
0094     Execute:= axisLinkInt.bResetAxis OR bResetAllAxis AND NOT E_STOP,
0095     Axis:= Axis,
0096     Done=> ,
0097     Busy=> ,
0098     Error=> ,
0099     ErrorID=> );
0100
0101 MoveAbsolute(
0102     Execute:= axisLinkInt.bMoveAbsolute AND NOT E_STOP AND axisLinkExt.bCalibrated,
0103     Position:= axisLinkInt.lrTarget,
0104     Velocity:= lrVelocity,
0105     Acceleration:= lrAcceleration,
0106     Deceleration:= lrDeceleration,
0107     Jerk:= lrJerk,
0108     BufferMode:= ,
0109     Options:= ,
0110     Axis:= Axis,
0111     Done=> ,
0112     Busy=> ,
0113     Active=> ,
0114     CommandAborted=> ,
0115     Error=> ,
0116     ErrorID=> );
0117
0118 ManualMove(
0119     JogForward:=axisLinkExt.bManForward AND NOT E_STOP,
0120     JogBackwards:= axisLinkExt.bManBackward AND NOT E_STOP,
0121     Mode:= MC_JOGMODE_CONTINOUS,
0122     Position:= ,
0123     Velocity:= lrActualManualSpeed,
0124     Acceleration:= lrAcceleration,
0125     Deceleration:= lrDeceleration,
0126     Jerk:= lrJerk,
0127     Axis:= Axis,
0128     Done=> ,
0129     Busy=> ,
0130     Active=> ,
0131     CommandAborted=> ,
0132     Error=> ,
0133     ErrorID=> );
0134
0135 Halt(
0136     Execute:= E_STOP OR axisLinkInt.bStopMovement,
0137     Deceleration:= 15,
0138     Jerk:= 15,
0139     BufferMode:= ,
0140     Options:= ,
0141     Axis:= Axis,
0142     Done=> ,
0143     Busy=> ,
0144     Active=> ,
0145     CommandAborted=> ,
0146     Error=> ,
```

```
0147        ErrorID=> );
0148
0149 IF bAddToGroup AND axisLinkExt.bCalibrated THEN
0150        bRemoveFromGroup := FALSE;
0151        bAddAxisToGroup := TRUE;
0152 END_IF
0153
0154 axisCouplingVisR_TRIG(CLK:= bAxisCouplingVis , Q=> );
0155
0156 IF axisCouplingVisR_TRIG.Q AND bAddAxisToGroup THEN
0157        bAddAxisToGroup := FALSE;
0158        bRemoveFromGroup := TRUE;
0159 ELSIF axisCouplingVisR_TRIG.Q AND NOT bAddAxisToGroup THEN
0160        bRemoveFromGroup := FALSE;
0161        bAddAxisToGroup := TRUE;
0162 END_IF
0163
0164 AddAxisToGroup(
0165        bExecute:= bAddAxisToGroup,
0166        nGroupId:= axisLinkExt.uiGroupId,
0167        nAxisId:= Axis.NcToPlc.AxisId,
0168        nIndex:= Axis.NcToPlc.AxisId - 1,
0169        tTimeOut:= t#1s,
0170        bBusy=> bBusy,
0171        bErr=> ,
0172        nErrId=> );
0173
0174 ReconfigGroup(
0175        bExecute:= bRemoveFromGroup,
0176        nGroupId:= axisLinkExt.uiGroupId,
0177        tTimeOut:= t#1s,
0178        bBusy=> bBusy,
0179        bErr=> ,
0180        nErrId=> );
```

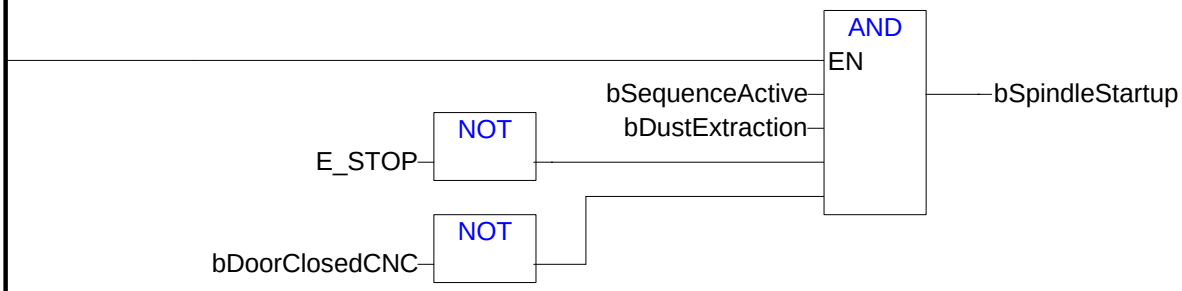
```
0001 PROGRAM CNC
0002 VAR_INPUT
0003     E_STOP: BOOL;
0004     bLoadProgramVis: BOOL;              (*Input from visualisation to load program*)
0005     bDoorClosedCNC: BOOL;               (*FALSE when door is closed*)
0006     bVacuumtableActivate: BOOL;         (*Input to activate vacuum table*)
0007     bStartCNCProgram: BOOL;             (*Input to start CNC program*)
0008     bXNulpunt: BOOL;                    (*Input to determine if machine is in starting/pickup position*)
0009     bYNulpunt: BOOL;                    (*Same as before*)
0010     bZNulpunt: BOOL;                    (*Same as before*)
0011     bRobotStartCNCProgram: BOOL;        (*Input from robot to start CNC program*)
0012     bRobotActiveInCNC: BOOL;            (*TRUE when robot is entering CNC machine, safety*)
0013     bRobotNotMoving: BOOL;              (*TRUE when robot is not moving, safety to open door*)
0014     bResetAllAxis: BOOL;                (*Resets all axis*)
0015     bGroupAxis: BOOL;                (*Groups axis, used to set spindle override*)
0016 END_VAR
0017 VAR_OUTPUT
0018     bOpenDoor: BOOL;                 (*Opens CNC door*)
0019     bCloseDoor: BOOL;                (*Closes CNC door*)
0020     bSpindleOut: BOOL;               (*Spindle control*)
0021     bDustExtraction: BOOL;              (*Starts vacuum cleaner*)
0022     bVacuumTable: BOOL;                 (*Starts vacuumtable*)
0023     bProgramFinished: BOOL;             (*TRUE if CNC program finished*)
0024 END_VAR
0025 VAR
0026     Channel2In   AT %I*:        NciChannelToPlc;        (* NC Channel inputs from PLC*)
0027     Channel2Out     AT %Q*:        NciChannelFromPlc;     (* NC Channel outputs to PLC*)
0028
0029     loadProg: ItpLoadProgEx;            (*FB to load program*)
0030     startProg: ItpStartStopEx;          (*FB to start program*)
0031     resetChannel: ItpResetEx2;          (*FB to reset channel after error*)
0032     EmergencyStop: ItpEStopEx;          (*FB to send E_STOP*)
0033
0034     spindleTON: TON;                    (*Delay timer for spindle, otherwise it might overload the circuit*)
0035     programmeTON: TON;                  (*Delay timer for program, to make sure spindle is up and running*)
0036     programOnTimerTON: TON;          (*Minimum time before program can be finished*)
0037     programFinishedTON: TON;         (*Delay after program is finished, safety*)
0038     bLoadDelay: TON;                 (*Delay before starting rest of sequence after program load*)
0039
0040     openDoorR_TRIG: R_TRIG;          (*Trigger to open door*)
0041     closeDoorR_TRIG: R_TRIG;         (*Trigger to close door*)
0042
0043
0044     bBusy: BOOL;                        (*Busy signal*)
0045     bLoadProgram: BOOL;                 (*Starts program load*)
0046     bStartSequence: BOOL;            (*Starts sequence after program load*)
0047     bSequenceActive: BOOL;              (*TRUE while sequence is active*)
0048     bStartProgram: BOOL;                (*Used to start the program*)
0049     bSpindleStartup: BOOL;           (*Used to start sequence of spindle startup*)
0050     bResetAfterE_STOP: BOOL;            (*Reset after E_STOP from visualisation*)
0051     bEndProgram: BOOL;                  (*Shuts down CNC machine after program is finished*)
0052     bResetCNC: BOOL;                 (*Resets CNC machine*)
0053     startTON: TON;                   (*Delay to make sure no accidental button pushes initiated program*)
0054     bStartCNC: BOOL;                    (*Used to start sequence*)
0055 END_VAR
```

0001

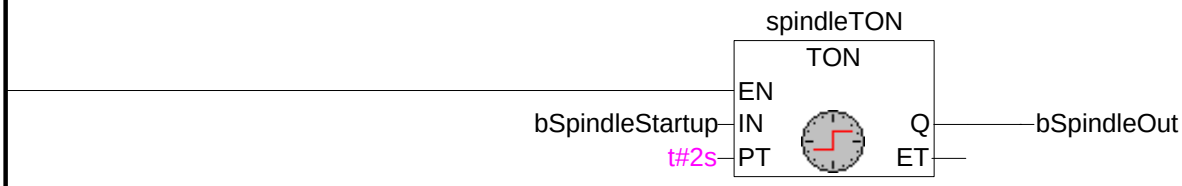



0002
bStartCNC
bRobotActiveInCNC
bVacuumtableActivate
E_STOP
bLoadProgram
0003
bVacuumtableActivate
bVacuumTable
0004
bGroupAxis
ItpSetOverridePercent
EN
100
fOverridePercent
Channel2Out
sPlcToNci
bBusy
0005
closeDoorR_TRIG
R_TRIG
EN
bLoadProgram
CLK
Q
bCloseDoor
0006
loadProg
ItpLoadProgEx
EN
bLoadProgram OR bLoadProgramVis
bExecute
bBusy
bBusy
'stadion.nc'
sPrg
bErr
12
nLength
nErrId
t#500ms
tTimeOut
Channel2In
sNciToPlc
0007
bLoadDelay
TON
EN
bLoadProgram
IN
Q
bStartSequence
t#1s
PT
ET
0008
bStartSequence
E_STOP
bRobotActiveInCNC
bProgramFinished
bOpenDoor
bSequenceActive
bLoadProgram
bDustExtraction
bSequenceActive

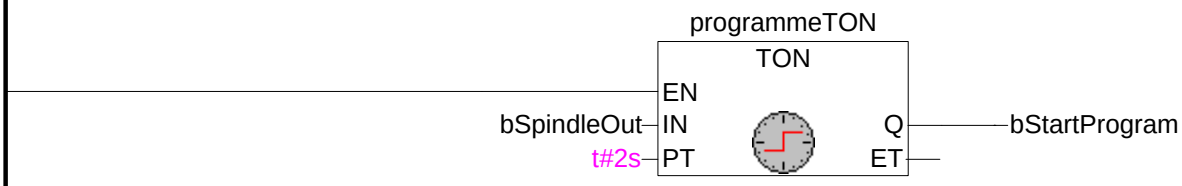
0009



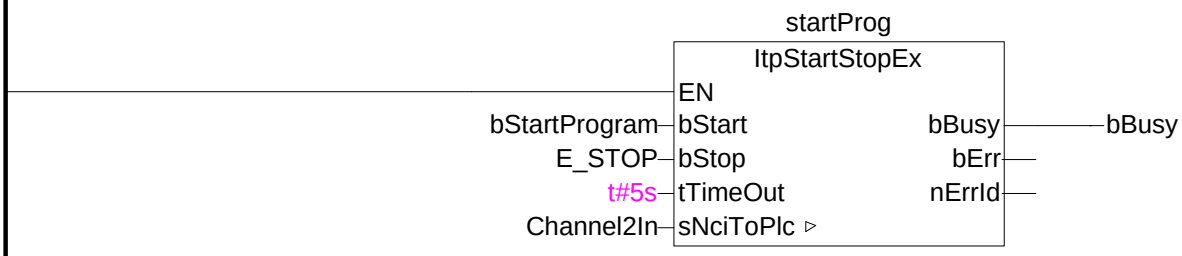
0010



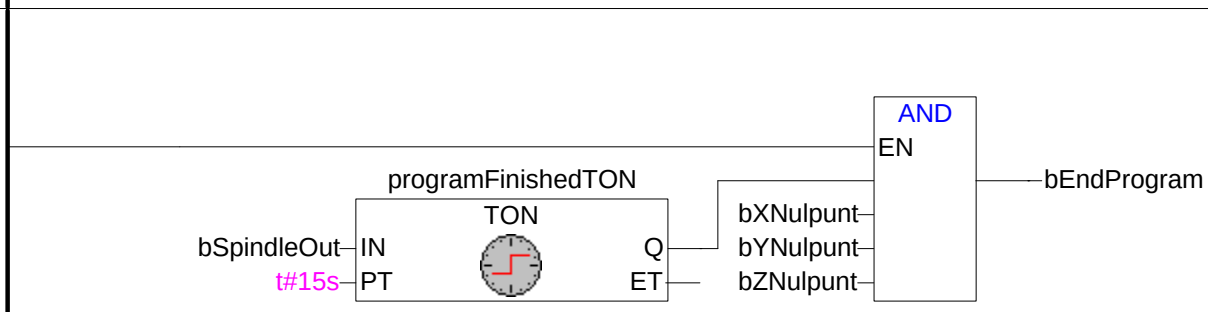
0011



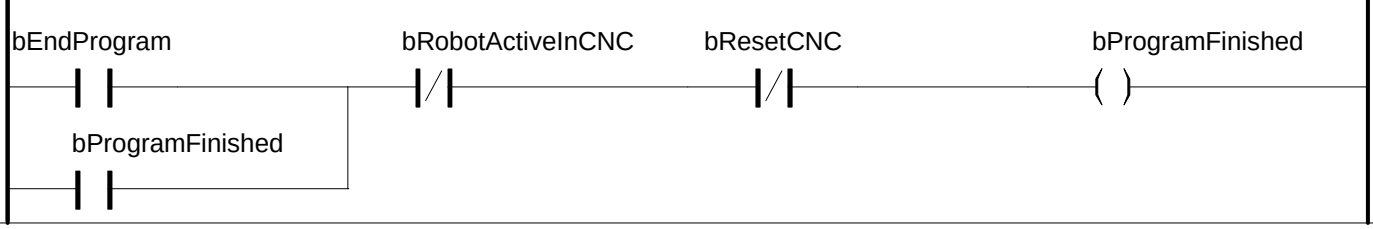
0012



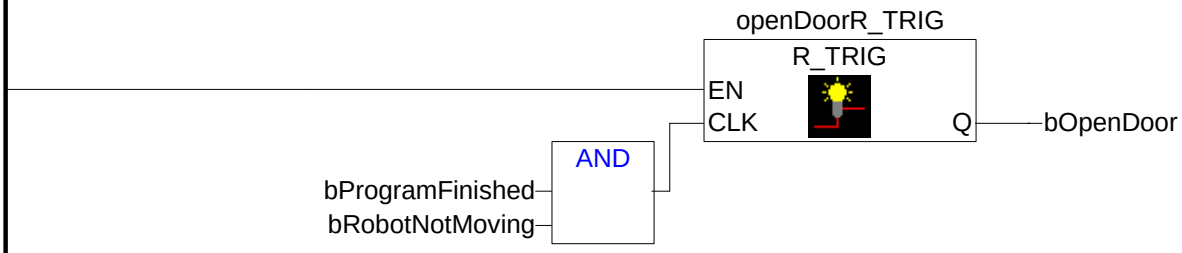
0013



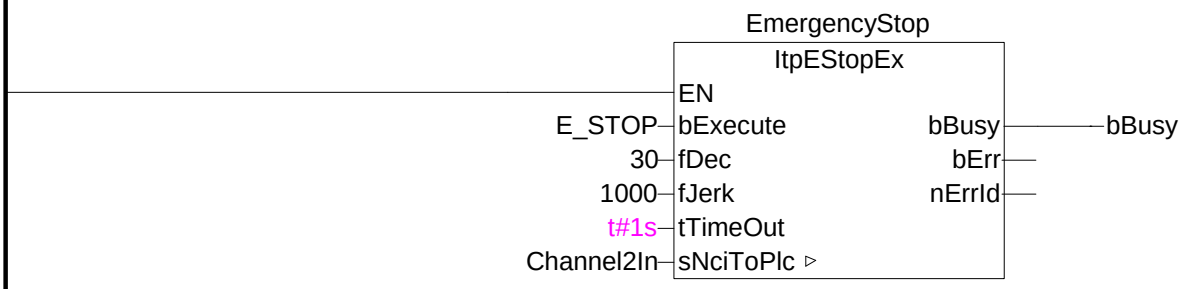
0014



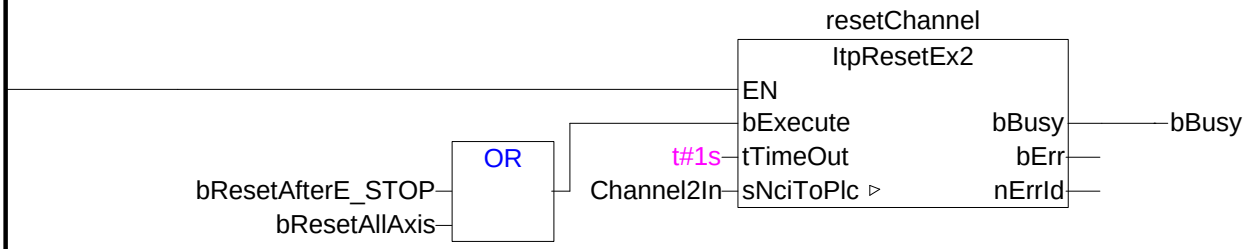
0015



0016

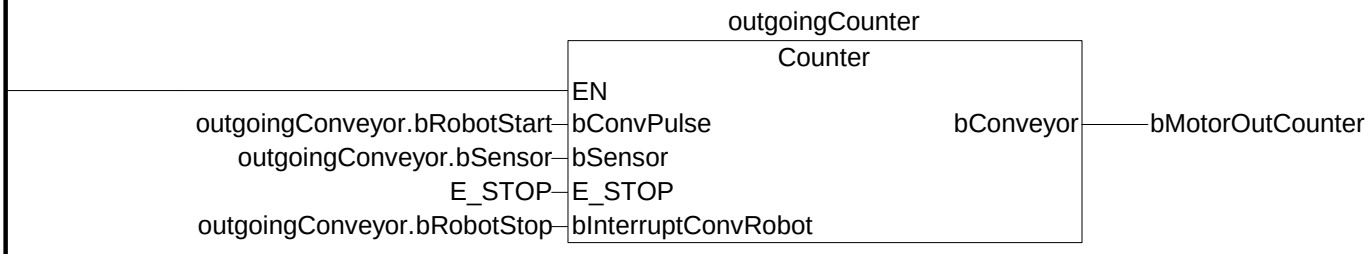


0017



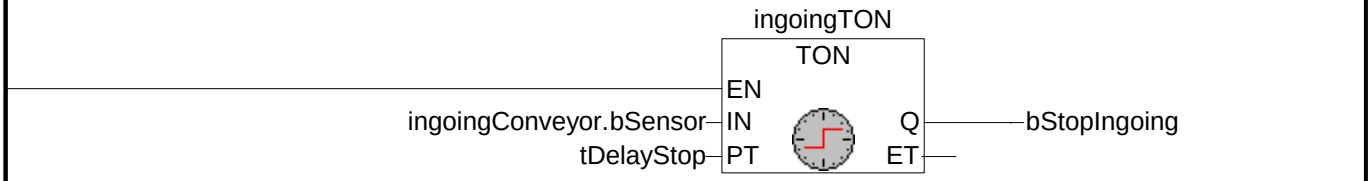
```
0001 PROGRAM Conveyors
0002 VAR_INPUT
0003     E_STOP: BOOL;
0004     bVisualisationPause: BOOL;      (*Pauses intake*)
0005 END_VAR
0006 VAR_OUTPUT
0007     bReadyForPickup: BOOL;          (*TRUE when block is ready for robot*)
0008 END_VAR
0009 VAR
0010     ingoingConveyor AT %I*: conveyor;   (*struct*)
0011     outgoingConveyor AT %I*: conveyor; (*struct*)
0012     bMotorIn AT %Q*: BOOL;          (*Motor output*)
0013     bMotorOut AT %Q*: BOOL;         (*Motor ouptut*)
0014     bStopIngoing: BOOL;             (*Stops ingoing conveyor*)
0015
0016     readyForPickup: TON;            (*Delay before pickup*)
0017     ingoingTON: TON;                (*Delay before motor is disabled*)
0018
0019     tDelayStop: TIME := t#250ms;    (*Delay before motor is disabled*)
0020     tDelayPickup: TIME := t#200ms;  (*Delay before pickup*)
0021     outgoingCounter: Counter;       (*Counts amount of blocks on outgoing*)
0022     bVisualisationMotorIn: BOOL;    (*Visualisation override ingoing*)
0023     bMotorOutCounter: BOOL;         (*Visualisation counter override*)
0024     bMotorOutVisualisation: BOOL;   (*Visualisation override outoing*)
0025 END_VAR
```

0001

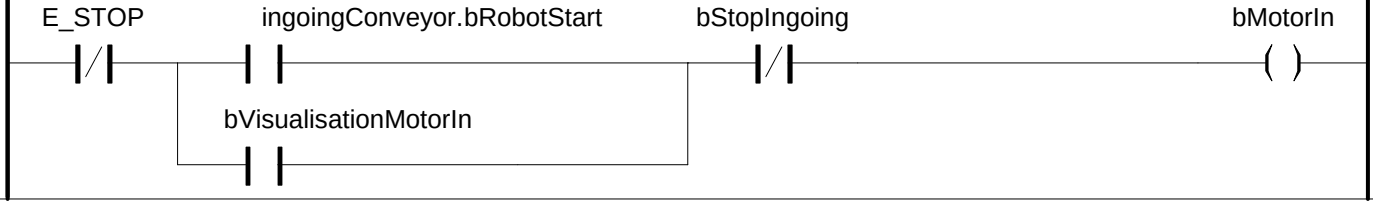


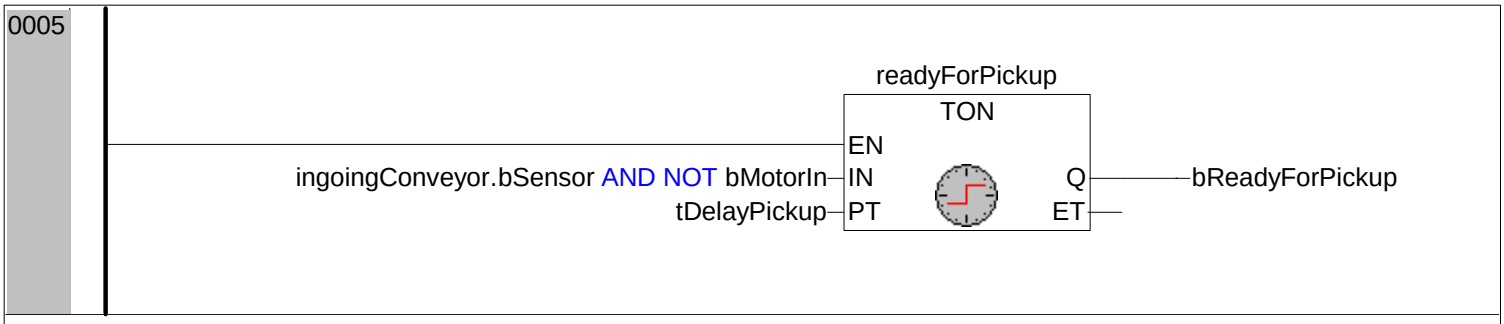
0002

0003



0004




0005
readyForPickup
TON
EN
ingoingConveyor.bSensor AND NOT bMotorIn
IN
tDelayPickup
PT
Q
ET
bReadyForPickup

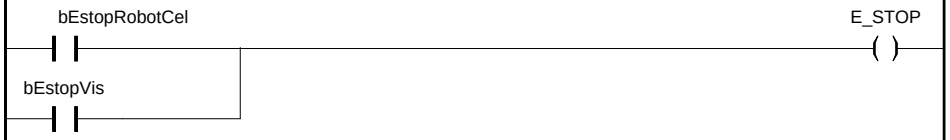
```
0001 FUNCTION_BLOCK Counter
0002 (*-----------------------------------------------------------INPUTS-----------------------------------------------------------*)
0003 VAR_INPUT
0004     bConvPulse: BOOL;              (*Ingang hoog indien de robot de band aanstuurt*)
0005     bSensor: BOOL;                     (*Ingang hoog indien er iets voor de betreffende sensor ligt*)
0006     E_STOP: BOOL;                      (*Emergency Stop*)
0007     bInterruptConvRobot: BOOL;     (*Pauses conveyor so the robot can place a new block*)
0008 END_VAR
0009 (*-----------------------------------------------------------OUTPUTS----------------------------------------------------------*)
0010 VAR_OUTPUT
0011     bConveyor: BOOL;                   (*Uitgang waarmee de band gestuurd wordt*)
0012 END_VAR
0013 (*-----------------------------------------------------------INTERNAL---------------------------------------------------------*)
0014 VAR
0015     conveyerRobotF_TRIG: F_TRIG; (*Trigger om de teller aan te sturen, signaal afkomstig van de robot*)
0016     sensorF_TRIG: F_TRIG;          (*Trigger om de teller aan te sturen, signaal afkomstig van de sensor*)
0017     loadedBlocksCTUD: CTUD;        (*Teller om het aantal voorwerpen bij te houden*)
0018     bCountUp: BOOL;
0019     bCountDown: BOOL;
0020     RESET: BOOL;                   (*Reset for testing and error correction*)
0021     iLoadedBlocks: INT;            (*Aantal stuks dat momenteel op de band ligt, om te bepalen of deze blijft lopen*)
0022     bCountUpVis: BOOL;
0023 END_VAR
```

```
0001 conveyerRobotF_TRIG(CLK:= bConvPulse OR bCountUpVis, Q=> bCountUp);
0002 sensorF_TRIG(CLK:= bSensor, Q=> bCountDown);
0003 loadedBlocksCTUD(
0004     CU:= bCountUp,
0005     CD:= bCountDown,
0006     RESET:= RESET,
0007     LOAD:= ,
0008     PV:= 10,
0009     QU=> ,
0010     QD=> ,
0011     CV=> iLoadedBlocks);
0012
0013 IF iLoadedBlocks > 0 AND NOT E_STOP THEN
0014     bConveyor := TRUE;
0015 ELSE
0016     bConveyor := FALSE;
0017 END_IF
0018
0019 IF bInterruptConvRobot OR bSensor THEN
0020     bConveyor := FALSE;
0021 END_IF
```

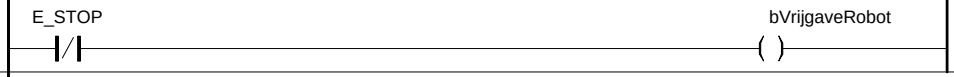
```
PROGRAM MAIN
VAR
(*----------INPUTS----------*)
    bEstopRobotCel AT %I*: BOOL;            (*External E_STOP from robot*)
    bDoorOpenCNC AT %I*: BOOL;              (*Input is FALSE when door is open*)
    bDoorClosedCNC AT %I*: BOOL;            (*Input is FALSE when door is closed*)
    bActivatePlaftorm AT %I*: BOOL;             (*Robot input to activate platform*)
    bActivateVacuumTable AT %I*: BOOL;      (*Robot input to activate vacuumtable*)
    zekeringen AT %I*: zekeringen;          (*Fuses*)
    bRobotNotMoving AT %I*: BOOL;           (*TRUE when robot is not moving*)
    bRobotStartCNC AT %I*: BOOL;            (*Input from robot to start CNC*)
    bRobotActiveInCNC AT %I*: BOOL;         (*TRUE when robot is enetering CNC machine*)
    bProgramRequestRobot AT %I*: BOOL;      (*Programnumber request from robot*)
    bRobotDrivesActive AT %I*: BOOL;        (*TRUE when drives are active*)
(*----------OUTPUTS----------*)
    E_STOP AT %Q*: BOOL;
    bSafeToEnterCNC AT %Q*: BOOL;           (*Output to robot, to make sure it is safe to enter the CNC machine*)
    bSpindleOut AT %Q*: BOOL;               (*Output to CNC spindle*)
    bVacuumTable AT %Q*: BOOL;              (*Output to vacuumtable*)
    bReadyForPickupIngoing AT %Q*: BOOL;        (*Output to robot when there is a block on the ingoing conveyor*)
    bDustExtraction AT %Q*: BOOL;           (*Output to vacuumcleaner*)
    bLEDs AT %Q*: BOOL;                     (*Output to LEDlights*)
    bSafeToEnterAfterCNC AT %Q*: BOOL;      (*TRUE when CNC is finished and it is safe to enter*)
    bStartRobot AT %Q*: BOOL;               (*Start robot program cycle*)
    bAandrijvingenRobot AT %Q*: BOOL;           (*Activates all robot drives*)
    bVrijgaveRobot AT %Q*: BOOL;
    bProgram1 AT %Q*: BOOL;                 (*Program numbers to send to robot, in case of fully AUTO_EXT use*)
    bProgram2 AT %Q*: BOOL;
    bProgram3 AT %Q*: BOOL;
    bProgram4 AT %Q*: BOOL;
    bProgramNumberCorrect AT %Q*: BOOL;         (*TRUE when program number is correct for robot*)

(*----------INTERNAL----------*)

    XaxisControl: AxisControl;
    YaxisControl: AxisControl;
    ZaxisControl: AxisControl;
    PneumaticControl: Pneumatics;

    bEstopVis: BOOL;

    bSafeToPickX: BOOL;
    bSafeToPickY: BOOL;
    bSafeToPickZ: BOOL;

    bCloseDoorCNC: BOOL;
    bOpenDoorCNC: BOOL;
    bGroupAxis: BOOL;

    bSafeToPickCNCDoor: BOOL;
    bVisualisationPause: BOOL;
    bSpindleStart: BOOL;
    spindleTON: TON;
    bCalibrateAll: BOOL;
    bActivateVacuumTableVis: BOOL;
    bSpindleStartVis: BOOL;
    bAddAxisToGroup: BOOL;
    bRemoveAxisFromGroup: BOOL;
    bActivateAllAxis: BOOL;

    bStartCNC: BOOL;
    bLoadProgramme: BOOL;
    bLoadProgramVis: BOOL;
    bCNCFinished: BOOL;
    bClampActive: BOOL;
    bResetAllAxis: BOOL;
    bAxisCouplingVis: BOOL;
```

```
0068    afterCNCTON: TON;
0069    bStartupRobot: BOOL;
0070    bDustExtractionCNC: BOOL;
0071    bDustExtractionVis: BOOL;
0072 END_VAR
```

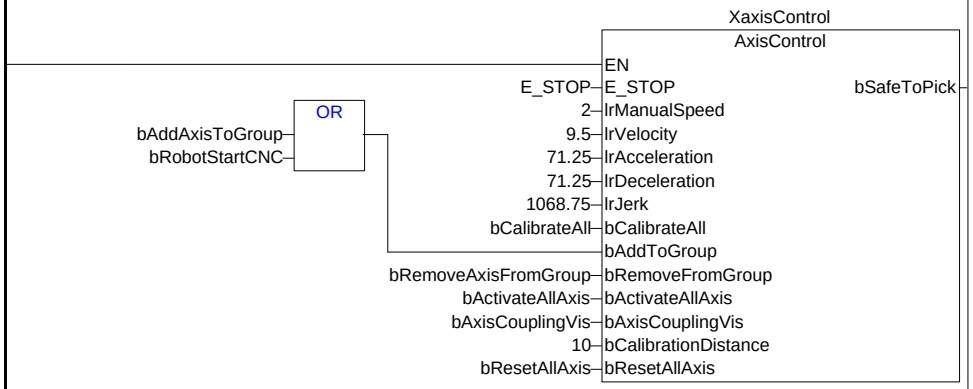
0001



0002

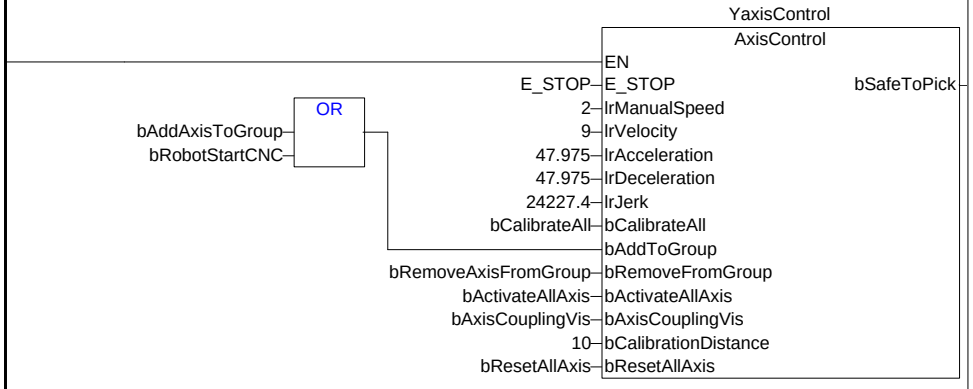


0003



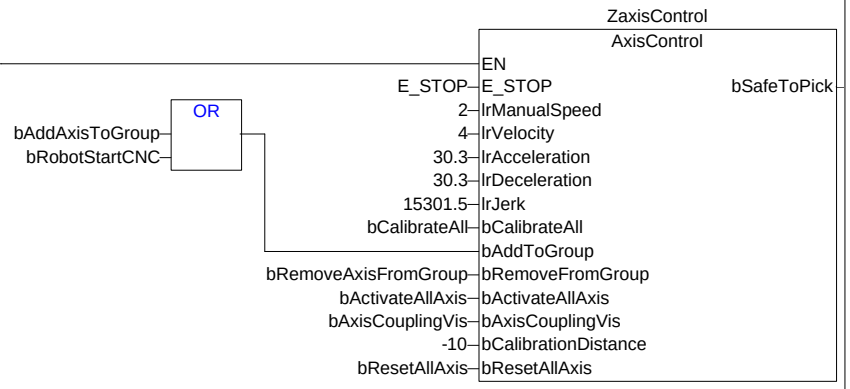
bSafeToPickX

0004



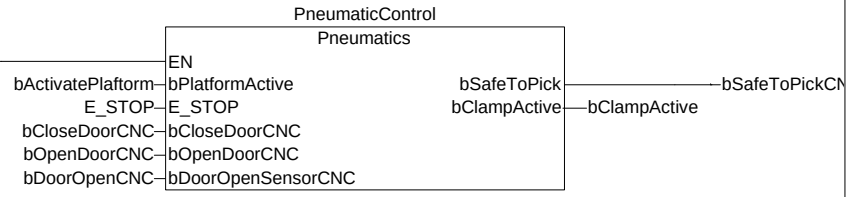
bSafeToPickY

0005



bSafeToPickZ

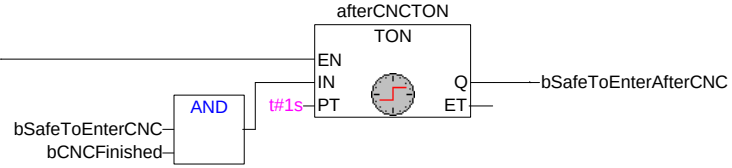
0006



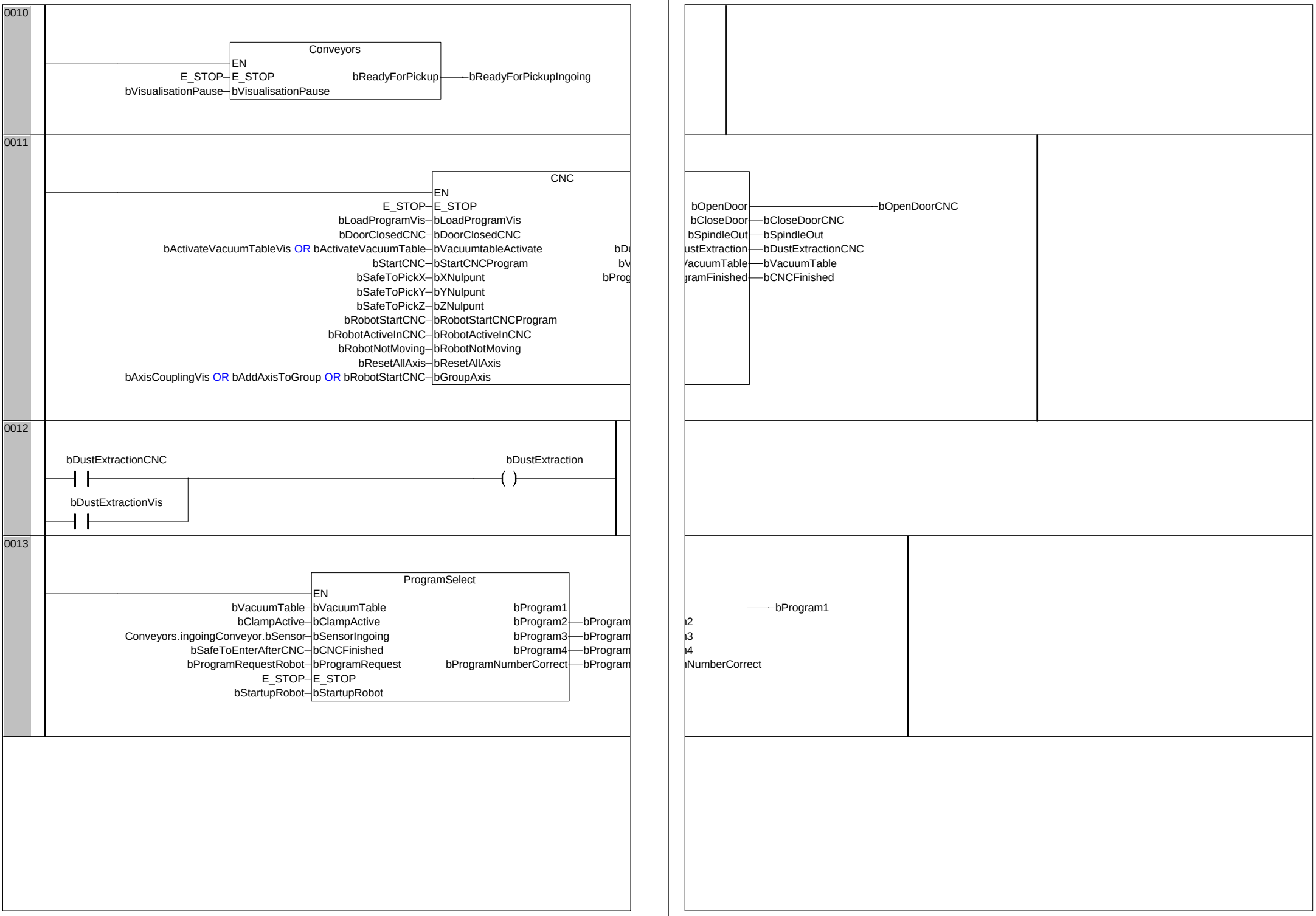
NCDoor

0007

0008



0009


0010
Conveyors
EN
E_STOP
E_STOP
bReadyForPickup
bReadyForPickupIngoing
bVisualisationPause
bVisualisationPause
0011
CNC
EN
E_STOP
E_STOP
bLoadProgramVis
bLoadProgramVis
bDoorClosedCNC
bDoorClosedCNC
bActivateVacuumTableVis OR bActivateVacuumTable
bVacuumtableActivate
bStartCNC
bStartCNCProgram
bSafeToPickX
bXNulpunt
bSafeToPickY
bYNulpunt
bSafeToPickZ
bZNulpunt
bRobotStartCNC
bRobotStartCNCProgram
bRobotActiveInCNC
bRobotActiveInCNC
bRobotNotMoving
bRobotNotMoving
bResetAllAxis
bResetAllAxis
bAxisCouplingVis OR bAddAxisToGroup OR bRobotStartCNC
bGroupAxis
bOpenDoor
bOpenDoorCNC
bCloseDoor
bCloseDoorCNC
bSpindleOut
bSpindleOut
bDustExtraction
bDustExtractionCNC
bVacuumTable
bVacuumTable
bCNCFinished
0012
bDustExtractionCNC
bDustExtraction
bDustExtractionVis
0013
ProgramSelect
EN
bVacuumTable
bVacuumTable
bClampActive
bClampActive
Conveyors.ingoingConveyor.bSensor
bSensorIngoing
bSafeToEnterAfterCNC
bCNCFinished
bProgramRequestRobot
bProgramRequest
E_STOP
E_STOP
bStartupRobot
bStartupRobot
bProgram1
bProgram2
bProgram3
bProgram4
bProgramNumberCorrect
bProgram1
bProgramNumberCorrect

```
FUNCTION_BLOCK Pneumatics
(*----------INPUTS----------*)
VAR_INPUT
    bPlatformActive : BOOL;                    (*Input vanaf robot*)
    E_STOP : BOOL;
    bCloseDoorCNC: BOOL;
    bOpenDoorCNC: BOOL;
    bDoorOpenSensorCNC: BOOL;
END_VAR
(*----------OUTPUTS----------*)
VAR_OUTPUT
    bSafeToPick: BOOL;
    bClampActive: BOOL;
END_VAR
(*----------INTERNAL----------*)
VAR
    clampBlockTON: TON;                    (*Timer gebruikt om de cilinders na elkaar aan te sturen*)
    tTimeActive: TIME;                     (*Tijdswaarde gebruikt om cilinders aan te sturen*)
    releaseBlockTON: TON;                  (*Timer gebruikt om klem te openen en alle outputs terug te resetten*)
    tOpen: TIME;                           (*Tijdswaarde gebruikt om outputs te resetten*)
    sideCilInR_TRIG: R_TRIG;               (*Triggers om de relais van de cilinders niet te overbelasten*)
    sideCilOutR_TRIG: R_TRIG;
    frontCilInR_TRIG: R_TRIG;
    frontCilOutR_TRIG: R_TRIG;
    closeDoorTOF: TOF;
    openDoorTOF: TOF;

    sideCilinder AT %Q*: pneumaticCilinder;
    frontCilinder AT %Q*: pneumaticCilinder;
    doorCilinder AT %Q*: pneumaticCilinder;
END_VAR
```

```
clampBlockTON(IN:= bPlatformActive AND NOT E_STOP, PT:= t#1s, Q=> , ET=> tTimeActive);
releaseBlockTON(IN:= NOT bPlatformActive AND NOT E_STOP, PT:= t#1s, Q=> , ET=> tOpen);
sideCilInR_TRIG(CLK:= , Q=> sideCilinder.bIngoing);
sideCilOutR_TRIG(CLK:= , Q=> sideCilinder.bOutgoing);
frontCilInR_TRIG(CLK:= , Q=> frontCilinder.bIngoing);
frontCilOutR_TRIG(CLK:= , Q=> frontCilinder.bOutgoing);


IF bPlatformActive THEN
    bClampActive := TRUE;
        IF tTimeActive > t#0ms AND tTimeActive <= t#50ms THEN
            frontCilOutR_TRIG.CLK := TRUE;
        END_IF

        IF tTimeActive > t#50ms AND tTimeActive <= t#100ms THEN
            frontCilOutR_TRIG.CLK := FALSE;
            frontCilInR_TRIG.CLK := TRUE;
        END_IF

        IF tTimeActive > t#100ms AND tTimeActive <= t#150ms THEN
            frontCilInR_TRIG.CLK := FALSE;
            sideCilOutR_TRIG.CLK := TRUE;
        END_IF

        IF tTimeActive > t#150ms AND tTimeActive <= t#200ms THEN
            frontCilOutR_TRIG.CLK := TRUE;
        END_IF

ELSE

        IF tOpen > t#0ms AND tOpen <= t#50ms THEN
            frontCilOutR_TRIG.CLK := FALSE;
            sideCilOutR_TRIG.CLK := FALSE;
            sideCilInR_TRIG.CLK := TRUE;
            frontCilInR_TRIG.CLK := TRUE;
        END_IF
```

```
0037
0038            IF tOpen > t#50ms AND tOpen <= t#100ms THEN
0039                sideCilInR_TRIG.CLK := FALSE;
0040                frontCilInR_TRIG.CLK := FALSE;
0041            END_IF
0042        bClampActive := FALSE;
0043    END_IF
0044
0045    IF E_STOP THEN
0046        frontCilOutR_TRIG.CLK := FALSE;
0047        sideCilOutR_TRIG.CLK := FALSE;
0048        sideCilInR_TRIG.CLK := FALSE;
0049        frontCilInR_TRIG.CLK := FALSE;
0050    END_IF
0051
0052    closeDoorTOF(IN:= bCloseDoorCNC, PT:= t#1s, Q=> doorCilinder.bIngoing, ET=> );
0053    openDoorTOF(IN:= bOpenDoorCNC, PT:= t#5s, Q=> doorCilinder.bOutgoing, ET=> );
0054
0055    IF NOT bDoorOpenSensorCNC THEN
0056        bSafeToPick := TRUE;
0057    ELSE
0058        bSafeToPick := FALSE;
0059    END_IF
```

```
0001 PROGRAM ProgramSelect
0002 VAR_INPUT
0003     bVacuumTable: BOOL;
0004     bClampActive: BOOL;
0005     bSensorIngoing: BOOL;
0006     bCNCFinished: BOOL;
0007     bProgramRequest: BOOL;
0008     E_STOP: BOOL;
0009     bStartupRobot: BOOL;
0010 END_VAR
0011 VAR_OUTPUT
0012     bProgram1: BOOL;
0013     bProgram2: BOOL;
0014     bProgram3: BOOL;
0015     bProgram4: BOOL;
0016     bProgramNumberCorrect: BOOL;
0017 END_VAR
0018 VAR
0019     iNextProgramNumber: INT;
0020     programTON: TON;
0021 END_VAR
```

```
0001 IF bCNCFinished THEN
0002     iNextProgramNumber := 3;
0003 END_IF
0004
0005 IF bClampActive AND NOT bVacuumTable AND NOT bCNCFinished THEN
0006     iNextProgramNumber := 2;
0007 END_IF
0008
0009 IF bSensorIngoing AND NOT bClampActive THEN
0010     iNextProgramNumber := 1;
0011 END_IF
0012
0013 IF NOT bClampActive AND NOT bVacuumTable AND NOT bCNCFinished THEN
0014     iNextProgramNumber := 1;
0015 END_IF
0016
0017 IF E_STOP THEN
0018     iNextProgramNumber := 0;
0019 END_IF
0020
0021 IF bStartupRobot THEN
0022     iNextProgramNumber := 4;
0023 END_IF
0024
0025 IF iNextProgramNumber = 1 THEN
0026     bProgram1 := TRUE;
0027     bProgram2 := FALSE;
0028     bProgram3 := FALSE;
0029     bProgram4 := FALSE;
0030 END_IF
0031
0032 IF iNextProgramNumber = 2 THEN
0033     bProgram1 := FALSE;
0034     bProgram2 := TRUE;;
0035     bProgram3 := FALSE;
0036     bProgram4 := FALSE;
0037 END_IF
0038
0039 IF iNextProgramNumber = 3 THEN
0040     bProgram1 := FALSE;
0041     bProgram2 := FALSE;
0042     bProgram3 := TRUE;
0043     bProgram4 := FALSE;
0044 END_IF
0045
0046 IF iNextProgramNumber = 4 THEN
```

```
0047        bProgram1 := FALSE;
0048        bProgram2 := FALSE;
0049        bProgram3 := FALSE;
0050        bProgram4 := TRUE;
0051    END_IF
0052
0053    programTON(IN:= bProgramRequest AND NOT E_STOP, PT:= t#500ms, Q=> bProgramNumberCorrect, ET=> );
```