

Fetstil är obligatoriskt, icke fet tillval
 < > runt teckenkombinationer
 variabel, programnamn o andra namn kursiva

Program:

Hantering av program, status, anropsträd, lista, gå in/ut ur program

Status/anropsträd:

STATUS visar status för alla program på stacken
STATUS tasknr visar status för angiven task, om tasken är abortad visas anropsträd med programnamn o stegnummer

Lista:

DIRECTORY listar alla program i minnet (RAM)
DIR/M listar alla modifierade program i minnet (RAM)
DIR/? listar alla felaktiga program i minnet (RAM)
DIR ro.* listar alla program som börjar på ro.
(LISTP program) listar angivet programs innehåll, ej speciellt användbart

Radera:

DELETEP prog1,prog2 raderar program **prog1** o **prog2** i minnet (RAM)
DELETE a.adv_cal raderar program **a.adv_cal** inkl. alla subrutiner o alla variabler som hör till programmet **a.adv_cal**

Avsluta/starta om:

ABORT tasknr stoppa program i task enl. tasknr.
RETRY tasknr starta program – startar med steget program stannat på
PROCEED tasknr starta program – startar med steget efter det program stannat på (rekommenderas ej!)
KILL tasknr tar bort task 0 eller angiven task från stack

Gå i/ur:

SEE program, stegnr gå in i program för att se eller redigera, ange stegnummer för att gå direkt till angivet steg
SEE om program stannat behöver ej programnamn anges
DEBUG tasknr gå in i debugger för att se/felsöka/redigera i angiven task
DEBUG om program stannat behöver ej programnamn anges
<F4> (Exit) gå ur editor(redigering)/debugger till monitor

I editor/debugger:

<F3> gå in i subrutin – cursor på rätt rad (CALL ro.sub)
<Shift+F3> gå ur subrutin till anropande program
<F11> växla från debug till monitor mode
<Shift+F11> växla från monitor till debug mode
<H> bläddra bland redigerade program
<Esc+H> gå till valt program
<Esc> välj COMMAND-mode
<I> välj INSERT-mode, infogningsläge (rekommenderas)
<R> välj REPLACE-mode, överskrivningsläge
<Lnn> gå till rad nn – i COMMAND-mode
<-nn Ctrl+Delete> återfå raderade rader **nn** stycken

Stega i debugger:

Placera cursor, i debugger, på rad att stega från, på skiljelinje visas aktuellt stegnummer;
XSTEP,,stegnummer ange steg för start stegning, OBS två komman!
<Ctrl+g> gå direkt till rad som cursor står på, enklare än XSTEP
<Ctrl+x> stega
<Ctrl+z> stega utan att gå in i subrutiner – hela CALL exekveras direkt
<Ctrl+b> infoga breakpoint på aktuell rad
<Ctrl+n> ta bort breakpoint på aktuell rad
BPT ta bort alla breakpoints i aktuellt program

Filer:

Hantering av filer, lista, kopiera, spara, ändra namn, flytta, kataloger, BAK-filer

Lista:

FDIRECTORY	listar alla filer o kataloger i defaultkatalog
FDI drive:\katalog\fil.ext	listar filer enl. specifikation
FDI drive:\katalog*.V2	listar alla filer med extension V2
FDI drive:\katalog*N*.*	listar alla filer som innehåller N
FDI *.PG	listar alla filer med ext. PG i defaultkatalog

Lista innehåll:

FLIST drive:\katalog\fil.ext	listar innehållet i filen enl. specifikation, används sällan
-------------------------------------	--

Vid kopiering, sparande och namnbyte får ej destinationsnamn finnas, tidigare befintlig fil måste raderas först!

Kopiera:

FCOPY drive:katalog\destfil. ext=drive:källfil.ext kopiera
FCOPY c:\applic\db-type.st=a:db-type.st kopierar filen **db-type.st** från diskett till katalog **applic** på hårddisk

Spara:

STORE drive:katalog\destfil.ext	sparar allt i RAM
STORE a:33050123	sparar allt i RAM på diskett, ext. blir .V2
STOREP a:33050123	sparar alla program i RAM på diskett, ext. blir .PG
STOREL a:33050123	sparar alla locations i RAM på diskett, ext. blir .LC
STOREP a:ro-main=ro.main	sparar alla program under ro.main på diskett, ext. blir .PG
STOREP/n a:ro-main=ro.main	sparar n nivåer under program ro.main på diskett, anges 1 sparas endast ro.main t.ex då ej subrutiner skall sparas

Byta namn:

FRENAME destfil.ext=källfil.st	döper om fil
FREN db-type.stb=db-type.st	döper om filen db-type.st till db-type.stb (BAK-fil)
FREN applic\main.pg=main.pgb	döper om filen main.pgb till main.pg i katalog applic

Radera:

FDELETE drive:\katalog\fil.ext	raderar fil
FDEL c:\applic\main.pg	raderar filen main pg i katalog applic på hårddisk (c:)
FDEL *.pg	raderar alla filer med extension .pg

Kataloger:

FDI/c drive:\katalog\	skapa katalog (/c = create)
FDI/c c:\kurs\	skapa katalogen kurs på hårddisk
FDI/d drive:\katalog\	radera katalog (/d = delete), katalog måste vara tom
FDI/d c:\kurs\	radera katalogen kurs på hårddisk
CD katalog	ändra defaultkatalog (C hange D irectory)
CD kurs	ändra defaultkatalog till kurs
DEFAULT D=katalog	ändra defaultkatalog
CD kurs	ändra defaultkatalog till kurs

Nätverk / TCP/IP / ethernet / NFS

NFS>	alltid före filbeteckningar på PC/Nätverk i övrigt som ovan (NFS=Net File Server), klarar långa filnamn
NFS>	ej AWC controllers (MV med nätkort, räcker ej med "N")
NET	nätverksstatus, IP-adress o mounts visas
PING aaa.aaa.aaa.aaa	kopplingskontroll nätverk, aaa... = IP-adress eller nodnamn
FSET tcp /NODE 'pc' /ADDRESS aaa aaa aaa aaa	sätt nod till pc med IP-adress aaa... , observera mellanslag ej punkt och inga ' för adress
FSET nfs /MOUNT 'xc' /NODE 'pc' /PATH 'C:\Adept\'	mount xc av katalog C:\Adept\ på nod pc

Variabler:

Hantering av variabler, lista, söka, sätta, lokala, globala, automatiska, in/ut, jokrar

Lista:

Alla:

LISTR	reals –tal
LISTS	strings – tecken
LISTL	locations – positioner

Vissa:

LISTR/LISTS/LISTL var_1,var_2,var_3...	lista en eller flera variabler (skilda med komma)
LISTR/LISTS/LISTL var[]	listar array
LISTR/LISTS/LISTL var[10]	listar arrayelement med index 10
DO @task TYPE variabel	(ej locations), automatiska kan förekomma i flera tasks

Med jokrar: (lämpligt när man inte vet exakt vad variabeln heter)

LISTR/LISTS/LISTL ro?	listar alla variabler som börjar med ro
LISTR/LISTS/LISTL ro?[]	listar alla enställda arrayvariabler som börjar med ro
LISTR/LISTS/LISTL ro?[,]	listar alla tvåställda arrayvariabler som börjar med ro
LISTR/LISTS/LISTL ro?[,,,]	listar alla treställda arrayvariabler som börjar med ro

Lokala/automatiska:

LISTR/S/L @ro.main	viss reservation för automatiska – endast om program på stack
	listar alla variabler i programmet ro.main

I debugger:

Dubbelklicka på önskad variabel, för uttryck dubbelklicka på parentes eller är-lika-med-tecken.

Sätta:

När man sätter variabler manuellt så förbigår man programmens inbyggda säkerhet, så det är mycket viktigt att man vet vad man gör! DO kan bara användas i abortad task, använd @task för att slippa aborta task 0 (default).

Reals (tal):

DO @task variabel=123	sätt variabel =talet 123
DO @task variabel=variabel_2	sätt variabel =värdet av variabel_2
DO @task var[1]=var[2]	kopiera värdet i var[2] till var[1] , gamla värdet i var[1] förloras

Strings (text/tecken):

DO @task \$variabel="text"	sätt \$variabel = text
DO @task \$string=\$name	sätt strängen \$string =innehållet i strängen \$name

Locations (positioner):

DO @task SET loc=TRANS(x,y,z,yaw,pitch,roll)	med koordinater som variabler
DO @task SET loc=TRANS(300,200,850,0,180,90)	med koordinater i siffror
DO @task SET loc=loc2	position loc =position loc2
DO @task SET loc=loc2:TRANS(300)	position loc =position loc2 förskjuten 300mm i x-led
HERE loc	sätt position loc till robotens nuvarande position
POINT loc	sätt ny position eller redigera befintlig

I debugger:

Klicka på önskad variabel, tryck <Shift-F5> ("TEACH"), ange nytt värde/text med "" (t.ex. "ny text")/locations med TRANS(x,y,z,p,r) (ändring av locations i debugger rekommenderas ej! använd POINT)

Radera:

DELETER	radera alla talvariabler (REALS) i RAM
DELETES	radera alla strängar (STRINGS) i RAM
DELETTEL	radera alla positioner (LOCATIONS) i RAM
DELETER/S/L var, var2	radera angivna/vissa variabler i RAM

Variabler: (forts.)

Det finns tre typer av variabeldefinitioner globala (default), lokala och automatiska. Globala och lokala variabler är odefinierade från början.

Globala:

Globala variabler skall endast finnas då de nyttjas av flera program t.ex. databaser (variantdata), eller för kommunikation mellan program. Globala variabler brukar ha samma prefix som de rutiner de styr t.ex. vi.--- för visionsrutiner eller db.--- för databaser.

Lokala:

Variabler blir lokala m.h.a. kommandot **LOCAL var1, var2...**

När ett program har exekverats finns de lokala variablerna kvar i minnet och kan alltid listas/användas igen. Lokala variabler är bra om man vill att ett program skall komma igång t.ex. vad som skedde förra gången det anropades. (motsvarar VB Static)

Automatiska:

Variabler blir automatiska m.h.a. kommandot **AUTO var1, var2...**

Automatiska variabler använder tillfällig plats i minnet och kan anta vilka värden som helst när de ej är satta av programmet och deras innehåll är endast giltigt om program ligger på stack.

Fördelen med automatiska variabler är att ett program kan exekveras i flera tasker utan att variablerna stör varandra mellan taskerna, för varje task reserveras eget minnesutrymme för de automatiska variablerna.

In o ut-variabler:

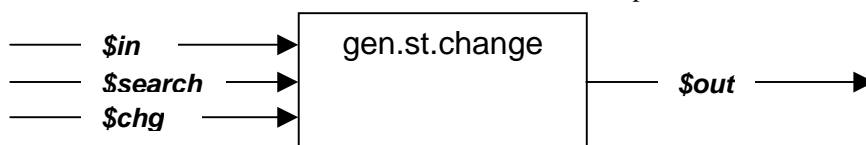
Vid programanrop kan in resp. utvariabler användas för att skicka variabler mellan anropande program o subrutiner. Subrutiner kan o bör ses som en "svart låda" där man skickar in data och får ut ett resultat.

In/utvariabler fungerar som automatiska variabler.

CALL gen.st.change(\$in,\$search,\$chg, \$out) program för att söka o ersätta text i strängar.

\$in anger text att ändra
\$search anger tecken/text att söka efter
\$chg anger tecken/text att ändra till
\$out anger ändrad text

I subrutiner finns variablerna beskrivna under INPUT PARM.: resp. OUTPUT PARM.:

**Signaler:****Växla status på signaler:**

SIGNAL signal, sig2
SIGNAL -q.ro.down
SIGNAL 2,-2003,f.quit

sätter signal, flera avdelas med komma
sätter signal **q.ro.down** låg – minus (-) framför
kan vara adress eller variabel

Se status på signaler:

LISTR SIG(i.ro.open)
IO
IO 1
IO 2

visar tillstånd på signalen **i.ro.open** (-1=ON – 0=OFF)
in- och utgångar
ingångar
flaggor
avsluta med <Ctrl+c>

För att se status på signaler är enklast att använda menyfunktioner.

Devicenet:

DEVICENET
DN.RESTART

listar noder och status på devicenet
startar om devicenet ifall t.ex. en nod varit bortkopplad eller felaktig

Devicenet-noder scannas och mappas i config_c.

Felsökning

Vid stopp:

Time-out eller Error-larm

Konrollera åtgärda

Öppna Monitor

Kontrollera/anteckna ev. felmeddelande:

*** Felmeddelande ***

Program task nn stopped at programn step nnn datum tid

Vanligaste felmeddelandena:

*** Undefined value ***

programmet har stannat pga en odefinierad variabel

*** Arithmetic overflow ***

division med noll, eller annan orimlig räkneoperation

Om felmeddelande försvunnit från monitor, t.ex. om man tittat i ett program kan **RET tasknr (RETRY)** användas för att visa meddelande igen.

eller...

Snurrar alla tasker **STAT (STATUS)** ev. kolla anropsstruktur **STAT tasknr**

Kontrollera om stoppad och i vilket program och vilket steg

Gå in i program **SEE programn, steg** anges **steg** ställer sig kursor på det steget direkt.

Olika fel i program:

*** Undefined value *** ta reda på vad variabeln heter, är den lokal eller global, var och hur definieras/sätts den, anteckna program stegnummer o variabelnamn och under vilka omständigheter felet uppkom, ta reda på vilket värde variabeln bör ha – använd debugger (lokal variabel) eller monitor (global) för att sätta variabeln till rätt värde.

Variabeln kan även sökas i program med **<F7> (FIND)** o **<F8> (REPEAT)** för att hitta var den definieras. Om arrayvariabel (t.ex. **\$db.var[i]**) kontrollera att pekare/index i hakar ([]) har rätt värde och definierad.

Väntar på en variabel (**WHILE, UNTIL, IF...GOTO**) program har ej stannat, använd debugger–stega igenom program och ta reda på varför program ej går vidare, vilka villkor uppfylls ej – vad heter variablerna i villkor vilket/vilka värden skall de ha – kommer variabler från annan task varför har den ej satt variabel till rätt värde kanske felet ligger där.

Skärmdump

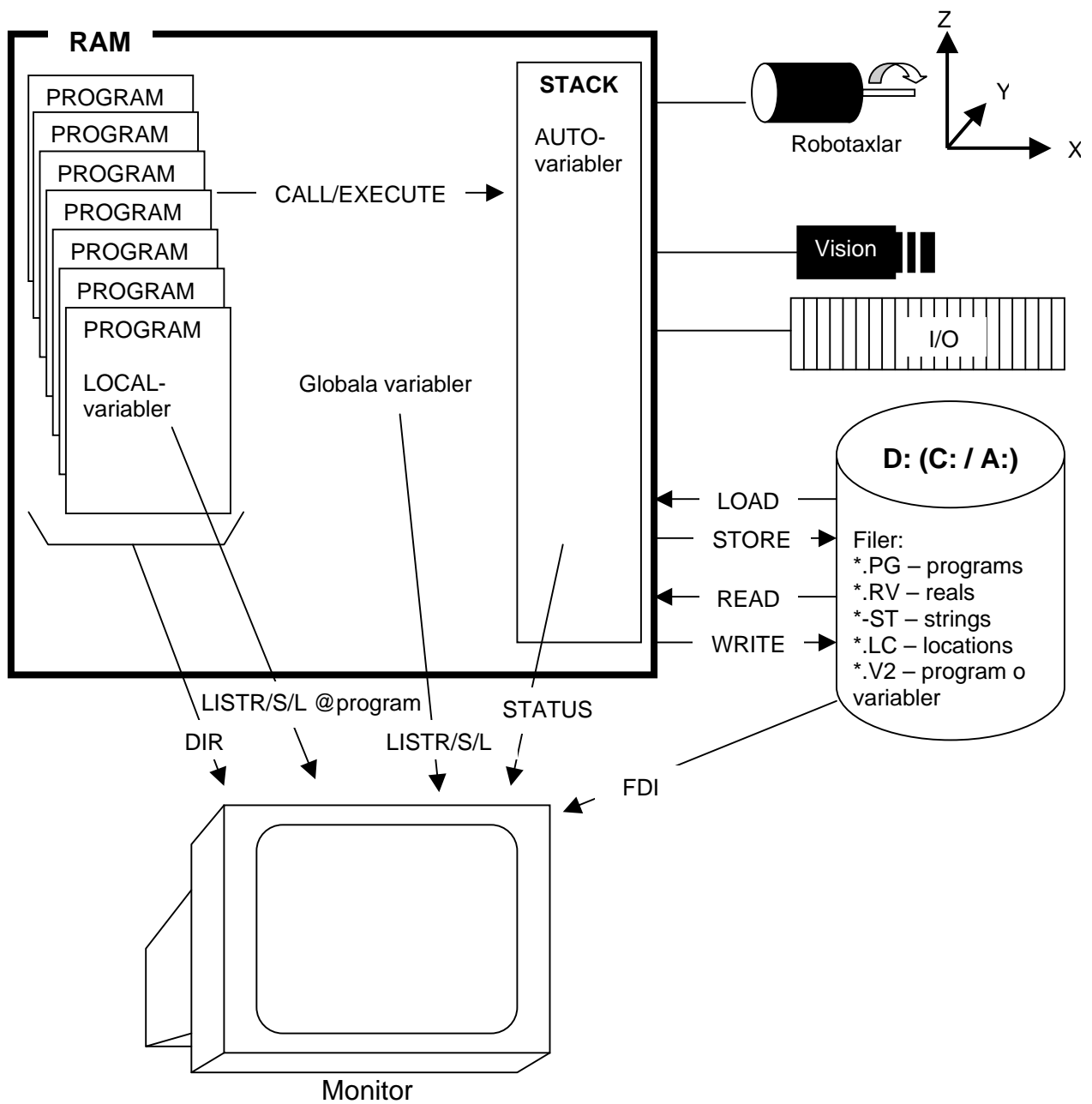
Vid fel som inte kan avhjälpas på ett enkelt sätt – som kräver omstart eller liknande – ta en "skärmdump" dvs. en bild av PC-skärmen med ev. larm eller statuslista. Se till att önskade delar av fönster syns på skärm! Visas något felmeddelande även i monitor – se till att det syns vid dump.

Tryck **<Print Scr>** (Print Screen) – starta Paint, klistra in och spara bild i lämplig katalog. Dessa bilder underlättar oftast för leverantör/programmerare att hitta och åtgärda orsak till felet. Skärmdumpar går bra att maila.

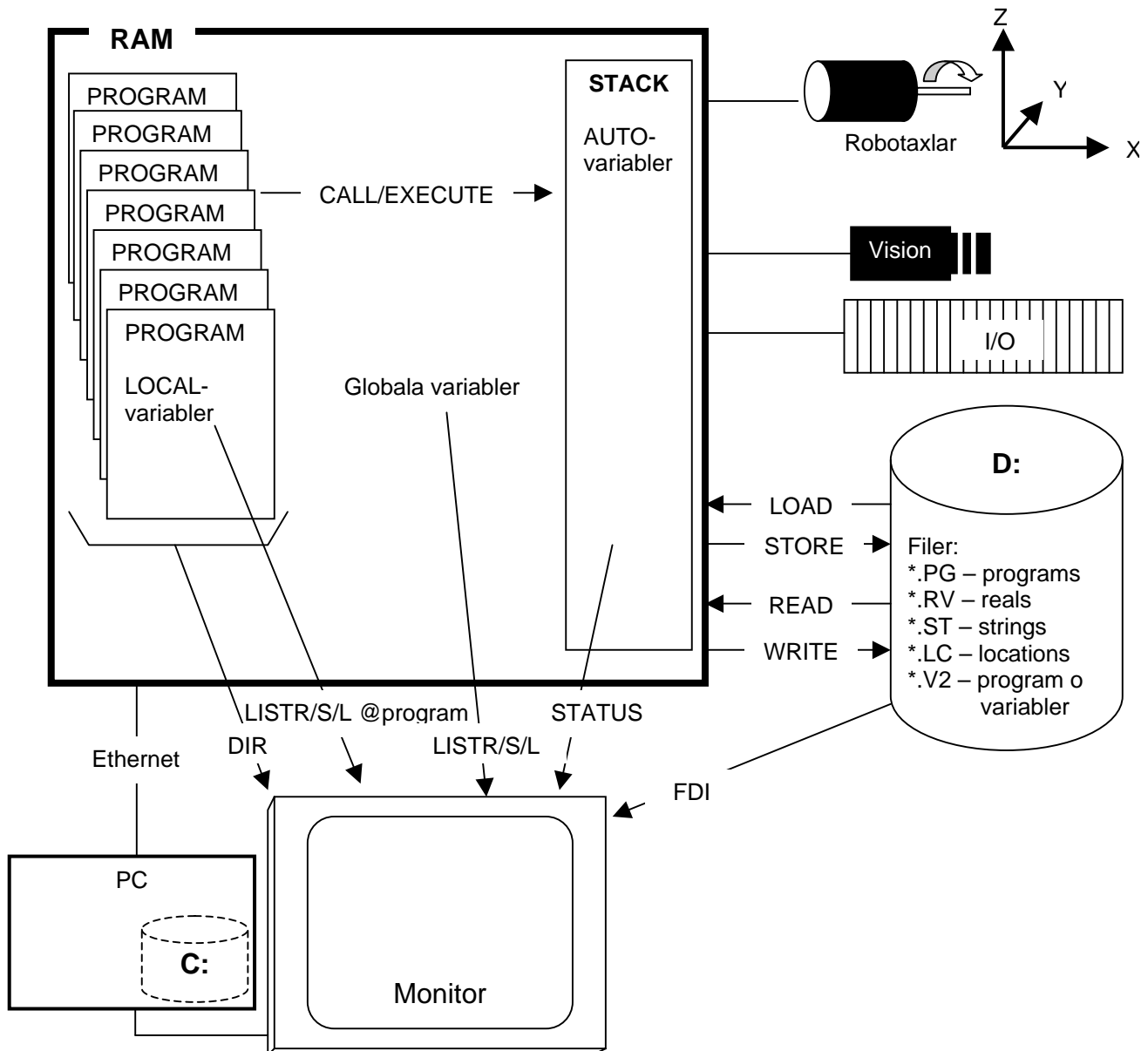
<Alt+Print Scr> sparar bara valt fönster.

Se även systemeringsdokument, programhuvuden och ev. readme-dokument för mer information om applikationens variabler.

Översikt MC/MV: variabler – program – filer – RAM – stack



Översikt AWC: variabler – program – filer – RAM – stack



Ordlista

array	en lista med variabler som kan vara en, två eller tredimensionell skrivs; "variabelnamn[x]", "variabelnamn[x,y]" resp. "variabelnamn[x,y,z]" , x/y/z är elementnummer i resp. dimension.
blob	ett område i en visionbild som ses som ett objekt.
digitala I/O	digitala in-/utgångar, signaler, används för kommunikation mellan system o styrning av ventiler (utgångar), kontroll av givare (ingångar) etc.
directory	motsvarande katalog, en undergrupp med filer, används för att strukturera filer på hårddisk o disketter.
edge	eng. kant, sida.
location	position, en lagrad punkt som roboten kan gå till eller använda i beräkningar av nya positioner.
pendant	programmeringslåda.
pixel	bildelement, de små ljuskänsliga halvledarelement som en visionkameran känslilyta("chip") är uppbyggd av, dessa omvandlar ljuset till en elektrisk signal som kan behandlas av styrsystemet. Mått på objekt i visionbilden anges oftast i millimeter men ibland i pixel. Vid kamerakalibrering så räknas förhållandet mm/pixel fram för att systemet skall kunna presentera mått i mm.
real	realvariabel, siffervariabel, innehåller ett numeriskt värde - tal.
realtid	att data visas i realtid betyder att aktuell data visas och uppdateras direkt när den ändras - direkt uppdatering och inte efter vissa perioder eller händelser.
string	strängvariabel, textvariabel, innehåller text, variabelnamn skrivs med \$-tecken först.
task	eng. uppdrag, i Adept-systemet kan flera program löpa samtidigt/parallell
variabel	sk. multitasking, de olika parallella processerna kallas för task(s). De numreras 0-27. ett register som kan ha ett valfritt namn och som kan anta olika värden eller innehålla text. ("komponenttyp ABC", "Kalle").