



This document will outline how to use the standard COM pseudocode (macros) in Winbuild 2000 to communicate to a DVT vision system. This document will often use examples pulled from the DVT documentation, and then show you how to form the same command/response with the Eason Winbuild 2000 software.

EXAMPLE FILE: "DVTvision.oip" example program provided with your installation of Winbuild2000. (Included in the "C:\programfiles\Winbuild2000v4\samples\" directory.)

▶ INITIALIZE COMMUNICATION

You will have to configure the Eason communication port with the proper parameters to share data with the DVT. Use the COM: INIT pseudocode available in the BASIC editor. The COM INIT setup should match the DVT camera settings. Put this code into the "On Open" of the first screen in your program, to be run just once in your program. Once initialized, the port will retain its configuration throughout the Eason program.

▶ SEND A COMMAND

To select product 1:

Command to DVT:

#PS 1 Select Product 1

Response:

% 0 Command completed without error.

? Command completed without error.

So the code in the Eason to send out this command, and read response back into a variable `string1$` (and also `string2$`)

```
COM: PUT COM 1 VALUE "#PS 1" RESPONSE string1$
COM: GET COM 1 TAG string2$
```

Then write a few lines of code to evaluate the responses from the DVT (`string2$`) and put it into a variable called `status$`:

```
REM: Now validate response from DVT
IF string2$ = "?" THEN
  status$ = "Command completed without error"
ELSE
  status$ = "Error performing command!"
ENDIF
```

▶ QUERY FOR INFORMATION

To query product 1 pass/fail stats:

Command to DVT:

#Pq 1 Query product 1 for pass/fail stats.

Response:

\$ 100 2 3 -1 Product 1 has performed 100 Pass, 2 Warn, 3 Fail
% 0 inspections, and the last inspection was Pass.

? command completed without error.

▼ READ RESPONSE

Thus the Eason will need to send a `#Pq 1 <CR>` and then read the responses back into three separate variables, and pick the relevant information apart.

```
COM: PURGE COM 1 BUFFER
COM: PUT COM 1 VALUE "#Pq 1" RESPONSE string1$
COM: GET COM 1 TAG string2$
COM: GET COM 1 TAG string3$
```

▼ EVALUATE RESPONSE

So the first variable `string1$` is going to contain your data, `string2$` and `string3$` are going to contain your responses from the DVT. You should always check `string2$` and `string3$` to make sure there were no errors or problems in the transmission or command:

```
IF string3$ = "?" THEN
  status$ = "Command completed without error"
ELSE
  status$ = "Error performing command!"
ENDIF
```

In this example, the error message was put into a string called `status$` for the interface to then display on screen to let the operator know there was a problem in communicating that command.

▼ EXTRACT INFORMATION

Pulling data from the `string1$` variable uses some simple string commands, and put into some separate numeric variables. Note that since we don't know how large the values are going to be, we have to find out where the spaces are, and then figure out how many characters are in each numerical position. Then pull out the relevant data:

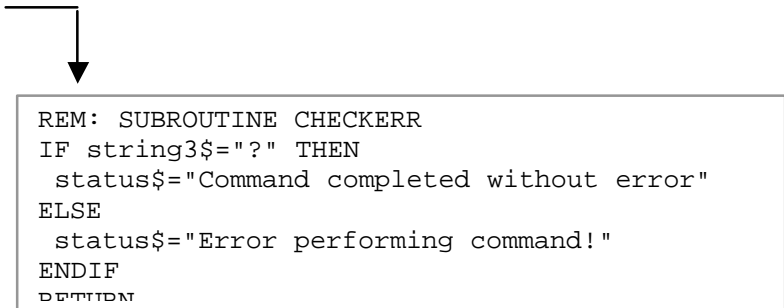
```
REM: This section figures out where each of the spaces are
WHERE1 = POS(string1$, " ", 3)
WHERE2 = POS(string1$, " ", (WHERE1+1))
WHERE3 = POS(string1$, " ", (WHERE2+1))

REM: This section pulls out each of the data points: pass, warn,
REM: fail, last value
prodlpass = VAL(MID$(string1$, 3, (WHERE1-3)))
prodlwarn = VAL(MID$(string1$, (WHERE1+1), (WHERE2-WHERE1)))
prodlfail = VAL(MID$(string1$, (WHERE2+1), (WHERE3-WHERE2)))
prodllost = VAL(RIGHT$(string1$, 2))
```

► OPTIMIZE

To optimize the code in your program, you will probably want to create some subroutines for commonly used code, like the verification code from above. Rather than placing the verification code a dozen times in the program every time you send a command out the port, create one subroutine and call the code whenever needed.

```
COM: PUT COM 1 VALUE "#Pq 1" RESPONSE string1$
COM: GET COM 1 TAG string2$
COM: GET COM 1 TAG string3$
GOSUB CHECKERR
```



```
REM: SUBROUTINE CHECKERR
IF string3$="?" THEN
  status$="Command completed without error"
ELSE
  status$="Error performing command!"
ENDIF
RETURN
```

▶ TRY IT OUT

Have the Eason unit powered up & available for download. (Safe Mode is fine.) Have your programming cable hooked up to the programming port on the Eason.

Click the “Compile, Download, Reboot!” button to send your program down to the unit.

After the unit has rebooted, try punching a few of your onscreen buttons and test communication to the DVT.