BEPI FOR PROGRAMMERS

General

This manual is intended for those with some experience of programming. Less experienced programmers may prefer to refer to a longer document “BEPI User Manual”.

As with any programming examples are invaluable and these can be downloaded from the EMACSYS web site at www.emacsys.com.

Introduction

BEPI (Basic EMACSYS Programmers Interface) is a simple programming language which enables the user to write applications which can interact with the physical I/O and other non-physical registers. While it capable of running quite sophisticated applications it kept intentionally simple and therefore it does not have the flexibility or complexity of other high level languages.

Because of its simplicity it is very easy to learn and hence well suited to those who have limited or no previous experience of programming.

BEPI Processing

The BEPI program is compiled and then downloaded into the EMACSYS. The appware will then run the compiled code. When it comes to the end of the code it will then return to its main tasks. Once those are complete it will again run the BEPI code and so on.

The period between each run will depend on how much time the main code needs.

Program Structure

The structure of the code is as follows:

```
Variables
/List of variables
End

Start
/Main program
End

sub Name
/Subroutine
subend

sub Name1
/Subroutine
subend
```

The first section defines the type and name of all the variables which are to be used in the program. Next is the main body of the program. Finally is the subroutines which will be used by the main part of the program.

Most applications will need to use some variables but for some simple applications they may not be necessary. In these cases the variables section can be left out.

All applications will need to use the main program section as this is where the code is run.

Variables

Variables must start with a letter but may contain numbers. No other characters such as under score are allowed in variable names. All variable names are case sensitive.

All variables are have global scope and must be declared in the list of variables.

Only unsigned variables are used and these are:

u8 is 8 bits wide (1 byte) and can hold values in the range 0 to 256.
u16 is 16 bits wide (2 bytes) and can hold values in the range 0 to 65535.
u32 is 32 bits wide (4 bytes) and can hold values in the range 0 to 4294967295.

Arrays

Only single dimension arrays are permitted.

Interface Variables

This type of variable is pre-defined and should not be included in the Variable section. Indeed if they are defined in the Variable section the compiler will generate an error.

These variables differ from the standard type in that they may be accessed and used by the main appware. For example “Inputs” will contain the current state of all the physical inputs. By reading this variable and after suitable processing writing to the variable “Outputs” it is possible to control the physical outputs.

As well as reading and writing to all the I/O in a block it is also possible read from or write to single elemets of the I/O. For example “Input1” will defines the state of input 1 while “Output2” defines the state of output 2.

As well as well as the physical I/O there is also a virtual variable. In a similar manner to the physical I/O this variable can be accessed as either a full variable or as individual elements. This can be used as a method for a user to alter the behaviour of the BEPI processing via the main appware. For example in the Ethernet Server it is possible to set a value in the virtual register. The BEPI processing can then inspect that value and act accordingly.

Code Elements

Assignments

Only the four arithmetic operations (+, -, * and /) are allowed in assignments.

```
a=b+c
a=b+c*d
```

Precedence of the arithmetic operations is in the order they are written. So in the above example c will be added to b and the result multiplied by d.

All numbers are base 10.

Conditional Statements

These take the form:

```
if Variable1=1
//Code
endif
```

or

```
if Variable1=1
//Code
else
//Code
```

endif

Only =, > and < allowed in the conditional test and these may be combinationed (e.g. >=) .

Loops

These take the form:

```
for 5
//Code
endfor
```

Each loop is repeated a fixed number of times. Variables are not allowed as an argument. Loops may be nested up to 5 deep.

Subroutines

Subroutines cannot have arguments and cannot return a value. They use global variables only and cannot have local variables.

Subroutines can only be called from the main code thus a subroutine cannot call another subroutine.

They take the form:

In the main code:

```
call Name
```

Then the subroutine:

```
sub Name
//Code
subend
```

Comments

Comments are denoted by a forward slash (/) as the first character on a line.

```
/This is a comment
        /This is not a comment and will generate an error
```

APPENDIX 1

INTERFACE VARIABLES

Inputs
Variable type u16.
Value of the physical inputs.
It is possible to write to this register but the new value will only be held until the inputs are read again at which point they will be updated to a value which reflects the value of the physical inputs. If the BEPI processing does change the value it will normally remain unchanged until just before the BEPI processing is applied again.
Read /Write

Input1
Input2
Input3
Input4
Input5
Input6
Input7
Input8
Input9
Input10
Input11
Input12
Input13
Input14
Input15
Input16
Variable type u8.
Value of the individual physical inputs. If the input is active it will read as 1 otherwise it will read as 0.
It is possible to write to these registers but the new value will only be held until the inputs are read again at which point they will be updated to a value which reflects the value of the physical inputs. If the BEPI processing does change the value it will normally remain unchanged until just before the BEPI processing is applied again.
Writing a 1 to the register will set the input to active. Writing a 0 to the register will set the input to inactive.
Read /Write

Outputs
Variable type u16.
Value of the physical outputs.
It is still possible for external parts of the appware to change the value of the output register. However in most applications these will not be sent to the physical outputs until after the BEPI processing.
Read /Write

Output1
Output2
Output3
Output4
Output5
Output6
Output7
Output8
Output9
Output10
Output11
Output12
Output13
Output14
Output15

Output16
Variable type u8.
Value of the individual physical outputs. If the output is active it will read as 1 otherwise it will read as 0.
It is still possible for external parts of the appware to change the value of the output register. However in most applications these will not be sent to the physical outputs until after the BEPI processing.
Writing a 1 to the register will set the output to active. Writing a 0 to the register will set the output to inactive.
Read /Write

Analogue1
Analogue2
Analogue3
Analogue4
Analogue5
Analogue6
Analogue7
Analogue8
Variable type u16.
Value of the physical A/D. The range of these registers will be 0 to 1023.
It is possible to write to these registers but the new value will only be held until the inputs are read again at which point they will be updated to a value which reflects the value of the physical inputs. If the BEPI processing does change the value it will normally remain unchanged until just before the BEPI processing is applied again.
Read /Write

Tick
Variable type u16.
This register is incremented every millisecond and has a maximum value of 65535. Once it has reached this value it will be reset to zero automatically. This register is useful for short timing period.
Note that it is not guaranteed that the BEPI processing will be applied within the 1mS time period. Therefore the value of this register may increase by more than 1 for each BEPI iteration.
Read /Write

Virtuals
Variable type u8.
Value of a non-physical register which is used for interaction between the external appware and the BEPI processing.
Read /Write

Virtual1
Virtual2
Virtual3
Virtual4
Virtual5
Virtual6
Virtual7
Virtual8
Variable type u8.
Value of a non-physical register which is used for interaction between the external appware and the BEPI processing.
Writing a 1 to the register will set the virtual output to active. Writing a 0 to the register will set the virtual output to inactive.
Read /Write

Seconds
Register containing the seconds value of the real time clock.
Read only.

Minutes
Register containing the minutes value of the real time clock.
Read only.

Hours
Register containing the hours value of the real time clock.

Read only.

Days
Register containing the day of the month value of the real time clock.
Read only.

Months
Register containing the month value of the real time clock. 1 = January..... 12 = December.
Read only.

Years
Register containing the year value of the real time clock.
Read only.

Weekday
Register containing the weekday value of the real time clock. 0 = Sunday, 1 = Monday ….. 7 =Saturday.
Read only.

APPENDIX 2

RESERVED WORDS

The following words cannot be used a variables:

return
if
else
endif
for
endfor
call
sub
subend