SOFTWARE

KR C

Submit Interpreter

Operation and Programming

Release 1

Issued: 04 May 2005 Version: 01

Contents

1 KR C and Submit interpreter

Two tasks run on the KR C controller:

- Robot interpreter (execution of robot motion programs)
- Controller interpreter (execution of a parallel control program)

1.1 Meaning and purpose of the controller interpreter

The controller interpreter can take over robot operator and control tasks. As this program runs entirely independently of the selected robot program, it can be used to handle all manner of different control tasks. These might include, for example, the control and monitoring of a cooling water circuit, the monitoring of safety equipment or the integration of additional peripheral devices. This renders the use of an additional PLC unnecessary for smaller applications as these tasks can be accommodated by the KR C.

1.2 Startup function

Depending on the specific application, the interpreter program can be started manually or automatically. The Startup function ensures that the Submit interpreter is started automatically each time the KR C robot controller is booted. The file entered as the start file in the `$custom.dat` file is started.

In the basic configuration of the software the program SPS.SUB is used
`$custom.dat` file: **`$PRO_I_O[ ]="/R1/SPS( )"`**

If the A10 arc welding equipment is being used, ARCSPS.SUB is entered here:
`$PRO_I_O[ ]="/R1/ARCSPS( )"`

1.3 What the Submit interpreter cannot do

The cycle time of the controller interpreter cannot be determined exactly as it must share the system performance of the KR C with the robot interpreter (which has a higher priority). This is why the controller interpreter is not suitable for time-critical applications; the outputs could have a reaction time of up to 50 ms! A separate PLC should be used in such cases.

For cases such as these we recommend use of the KUKA products "CoDeSys" or "Multiprog-ProCoNos". Detailed descriptions of these products can be found in the KUKA Roboter GmbH product catalog.

2 Programming the Submit interpreter

Submit files always have the file extension *.sub and are displayed exclusively in Expert mode. Program names for Submit files can consist of any characters up to a maximum length of eight characters.

A Submit file may only be viewed, created and edited at Expert level. The controller interpreter can be started and run at both user levels.

Unlike in the robot, program execution in the controller is carried out in quick cycles (PLC approach).

2.1 Expansion and modification of the SPS.SUB program

As the Submit file sps.sub is supplied as part of the basic configuration, it can be frequently modified and expanded by the user. If the control program in question is still running, it must first be stopped. It is then necessary to switch to Expert mode and load the desired control program into the editor. The modifications or expansions can now be carried out. The specially prepared program folds user.init and user.plc (open via the menu item “Act FOLD op”) should be used for this purpose.

The editor window must now be closed. This automatically starts the compiler. Once compilation has been successfully completed, the program is saved, following a request for confirmation, and is ready for execution. It is started by selecting it or via the menu item “Start Submit interpreter”.

Example sps.sub program

```
DECLARATIONS
INI
AUTOEXT INIT
ARC!= PLC INIT
USER INIT ;  <- Open this fold!
; Insert expansions for one-off settings following booting
; e.g. initialization of process variables
LOOP
AR10 PLC
TOUCH PLC
GRP PLC
SPOT PLC
USER PLC; <- Open this fold!
; Insert expansions for cyclical program execution
ENDLOOP
```

2.2 Creating and editing your own Submit program

To create a Submit file, it is necessary to switch first to Expert mode. All the loaded files are then displayed at the robot level. These are:

- Program files (extension *.src)
- Program data files (extension *.dat)
- Submit files (extension *.sub)

To create a new Submit file, press the softkey “NEW”. This generates an input box in which the name of the new Submit file and the extension *.sub must be entered. Once these entries have been made, the Enter key is pressed and the (at this stage still empty) Submit file is created. In order to enter commands in the Submit file, the file is highlighted using the blue bar and the editor is opened by pressing the softkey “EDIT”. The Submit file is now displayed and commands can be entered.

Program structure:

The structure is very similar to the program structure at the robot level. Data types, declarations and initializations are thus also handled in the same way as for a robot program.

Basic structure of a Submit program:

Declaration

Initialization

Program

2.3 Command set

Almost all KR C commands that have nothing to do with robot motions can be used in the Submit interpreter (exception: asynchronous motions can also be used in the Submit program). The robot interpreter has sole responsibility for the control of robot motions. This also means that a Submit program cannot call robot programs (`*.src`) as direct subprograms:

Permissible controller interpreter commands:

`LOOP`

`FOR loop`

`WHILE loop`

`REPEAT loop`

`IF branch`

`SWITCH statement`

`GOTO`

`EXIT` (to terminate a loop)

Serial interface commands

`READ / WRITE`

If you want controller interpreter programs to run permanently in the background, this must be programmed using a program loop (`Loop...Endloop`).

To prevent this cycle from being interrupted, `WAIT` commands and wait loops should not be programmed.

Example of the negative effects of a WAIT command:

```
Def PLC(  )
loop
$out[1] = true
wait sec 20
$out[2] = true
if $in[1] = = true then
$out[10] = true
endif
endloop
end
```

The WAIT command between the two output commands stops the program loop for 20 seconds. This means that the subsequent polling of the input $in[1] is no longer reliable, as this input is only polled briefly once every 20 seconds.

Commands which are only to be executed once after booting the system can be entered before the LOOP command. This allows basic initializations, for example, that are to take effect once the robot has been booted.

2.3.1 Access to system variables

The Submit interpreter can obtain read and write access to many system variables at any time, even if they are being used at the same time by a robot program.

2.3.2 Access to robot inputs and outputs

The controller interpreter can also access the inputs and outputs of the KR C, since they are nothing other than mapped system variables. As with the robot interpreter, the file `IOSYS.INI` is responsible for the external assignment of the inputs and outputs. Downstream bus systems can also be addressed simply via the controller interpreter.

2.4 Compiler

If new program code is entered in the editor, it must always be compiled. Only then can the code be run as a Submit program. The compiler is started automatically when the editor window is closed. All commands are checked and translated into machine commands. If the entered Submit program is free of errors, the Submit file is available and can be selected and started.

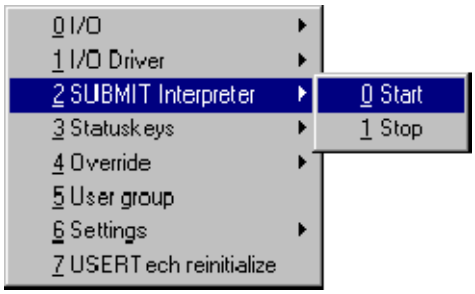
Incorrect entries: If the code contains incorrect entries, incompatible logic or basic motion command calls (PTP, LIN, CIRC), it cannot be translated by the compiler. In such cases an error message is generated in the user interface message window and an error file (*.err) is created. This file lists the errors that have been found along with their error number.

It is not possible to start an incorrectly compiled Submit file. Instead, the compiler generates an ERR file. This file may not be visible immediately in the listing, but then becomes visible following the command “Refresh display”.

3 Operator control

3.1 Manual start

The Submit interpreter can be started manually in the “Configure” menu. Following the manual start, the Submit program specified in the file `$custom.dat` is selected and executed independently of the status of the programs on the robot side of the KR C.



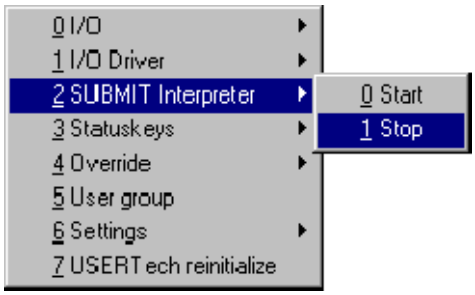
The **status flag** for the interpreter in the GUI turns green once the interpreter has started.

Start by means of manual selection:

Any available Submit program (and any robot program) can be selected using the bar and started by selecting the command “Select”. This procedure is thus independent of the file specified in `$custom.dat`.

3.2 Manual stop

The Submit interpreter can be stopped again manually via the menu: Configure–>SUBMIT Interpreter–>1 Stop. All activity of the interpreter is now stopped.



The **status flag** for the interpreter in the GUI turns red again, indicating that the interpreter has stopped.

3.3 Visualization of program execution for fault analysis

With the KR C in its initial state, the execution of a controller program is not displayed. Instead, according to what has been selected, the program window of the GUI displays either file directories or a selected robot program. In order to be able to view also the execution of a running Submit program, the system variable `$interpreter` can be set to the value “**0**” via the menu Monitor->Variable->Modify. The Submit program, complete with the main run pointer, is now visible in the program window.

It is not possible to switch this function back again in software version 2.3. To do this, the KR C must be rebooted.

From software version 3.2. onwards, the display can be switched back to the robot program by resetting the system variable to its original value “1”.

4 Examples

4.1 Calling subprograms

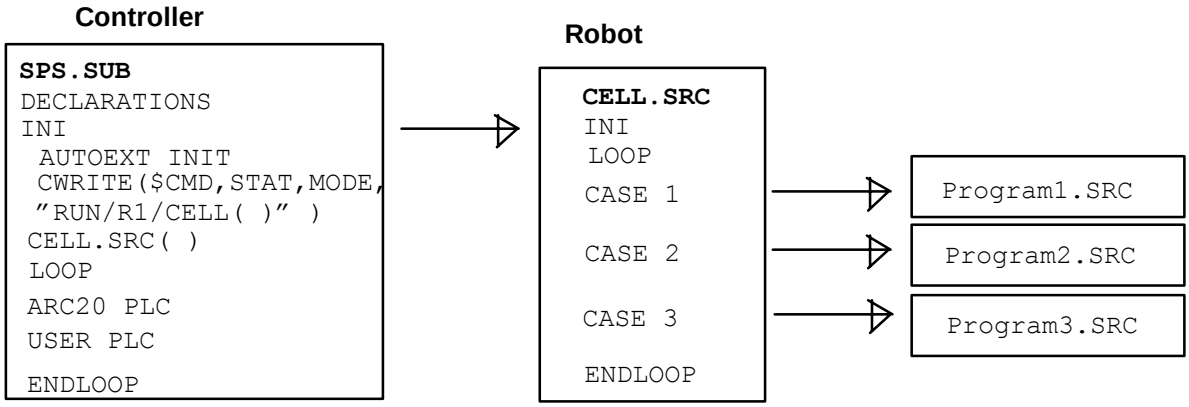
Controller subprograms (*.sub) can also be called from an interpreter program. The program call has the same syntax as that for a motion program in the robot interpreter.

Example program call of another Submit program:

```
Def Sub( )
ext unterpro( )
loop
unterpro( ) ; Calls the controller program unterpro.sub
endloop
end
```

Example of a robot program call from the Submit program:

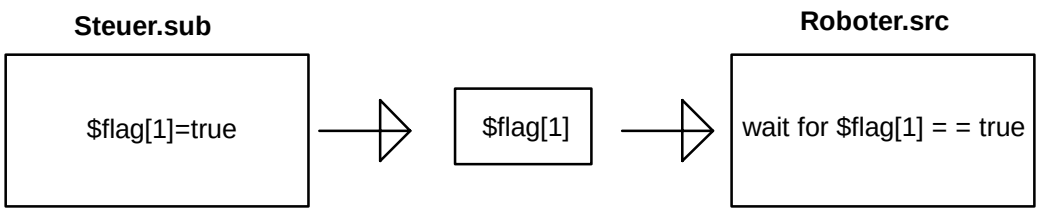
In this example, the Cell program of the robot interpreter is triggered by the initialization command **`Run Cell`** in the controller interpreter.



4.2 Communication with other applications

The KR C flags (notices) can be used to enable the exchange of binary information between a running robot program and an interpreter program. Any flag can be set by the controller interpreter and read by the robot interpreter. This allows the exchange of information between running programs and a certain degree of synchronization.

Principle of data exchange:



4.3 Controlling system variables

Flashing display

```
Def PLC(  )
$TIMER_STOP[10]=FALSE ;Timer(10) is started
loop
USER PLC
if $timer[10]>500 then
$phgcont=15 ;Set contrast to “15”
endif
if $timer[10]>1000 then
$phgcont=0 ;Set contrast to “0”
$timer[10]=0 ;Reset counter to “0”
endif
endloop
end
```

The program switches the contrast of the GUI screen between the maximum and minimum values using the system variable **$phgcont**. The time control functions are performed by Timer(10).

4.4 Polling system variables

Axis monitoring

```
Def PLC(  )
real Angle_A1
loop
USER PLC
Angle_A1 = $AXIS_ACT.A1
if ((Angle_A1 > -60.0) AND (Angle_A1 < 60.0)) THEN
$OUT[3] = True
$OUT[4] = True
else
$OUT[3] = False
$OUT[4] = False
endif
endloop
end
```

In this program, the system variable **$AXIS_ACT.A1** is polled and evaluated. In this way, the current axis position of A 1 can be determined by the Submit program at any time. As soon as A 1 moves outside the range +/– 60.0 degrees, the output variables $OUT[3] and $OUT[4] are set to “false”. The output variables can now be assigned through to the interfaces or be polled by the robot program and linked according to the specific application.

4.5 Access to inputs and outputs

Running light controller

```
Def PLC(  )
int b,a ;Declaration of variables b and a
$timer_stop[10] = false ;Start Timer(10)
b = 2
a = 1
USER PLC
loop
if $timer[10]>500 then
$out[b] = true ;Activate output (b)
$out[a] = false ;Deactivate previous output
b = b + 1
a = a + 1
$timer[10] = 0 ;Reset timer
endif
if b > 16 then ;Reset counter variable b
$out[a] = false ;Reset last output
b = 2
a = 1
endif
endloop
```

This program switches outputs 1 to 16 on and off, one after the other. The time control functions are performed by Timer(10). If the outputs are now viewed with the aid of the menu item “Overview output”, the effect is like that of a running light.

4.6 Technology packages

If the supplied ARC Tech technology package is to be integrated into the KR C software, the Submit program `arcsps.sub` must be selected first. This setting must be integrated into the Startup function if technology packages are used, otherwise the correct functioning of these program components cannot be assured.

Entry that must be made in the file `$custom.dat`: `$PRO_I_O[ ]=”/R1/ARCSPS( )”`

5 Useful information and tips

5.1 Runtime variable

Some variables cannot be accessed. Examples include the local variables of a robot program: so-called runtime variables.

5.2 Parallel activation of outputs

Particular care should be taken when activating outputs. The KR C compiler does not check, for example, whether the robot and controller interpreters are simultaneously accessing the same output. Parallel access by several programs (e.g. by the robot and controller interpreters simultaneously) to a single output may, in certain cases, be desired and is translated without errors by the compiler. As the robot and controller interpreters are not synchronized, this may result, in the case of incorrect programming, in undesired output signals at the KR C interfaces.

Careful attention must be paid when assigning the output variables. Since the output variables may be used to control machines, devices or safety equipment in the vicinity of the robot, an incorrectly set output can have fatal consequences.

Symbols

A

B

C

E

F

I

N

P

S

T

U

W