EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

Development tools for M-68000 based controllers in the PS Linac group.

PS/Linac Note 85-22

U. Raich

Geneva, 31 October, 1985

Abstract:

The Linac group plans to use the SMACC, a M-68000 based ACC, to perform ppm beamline switching. This project will very probably be followed by other control applications using the SMACC. It therefore seemed reasonable to invest some time on building up general tools supporting the development of application software for the SMACC. The cross-software available on the central machines was ported to our local Unix computer (our standard Q-bus equipment where the CPU card has been replaced by a PCS QU-68000). Thus the Modula-2 compiler, the Pascal compiler, the M68MIL assembler and all Cufom processors are now locally available. A special S-format loader converts S-format into binary code and loads it, via an RS 232 line, into the SMACC.

1. Introduction

In the Linac group a control facility to switch power supplies of a beamline on a ppm basis is currently being developed. During the Oxygen run, beginning in 1986, this facility will read the PLS protocol, containing information about the type of particle for the next 2 pulses coming from either of the two Linacs, and will switch the PS common beamline elements correspondingly. For reasons of economy and standardization it was decided to use the SMACC, a M-68000 based ACC (developed by W. Heinze PS-CO) for this purpose.

A large amount of cross-software is available from DD on the central machines. However these machines (especially the Priam Vax) are usually heavily loaded and not always easily accessible (we have no Index line in our Lab). It was therefore desirable to get the cross-software onto one of our local machines.

2. Cross-software on the PCS QU-68000

Our PDP-11 machines, having a total address range of 64 kbytes/task cannot be used as host machines for the cross-software (The M68MIL assembler uses 260 kbytes of executable code). However, since the beginning of this year our group has Q-bus computer with a MC-68000 CPU card from PCS Munich, running the UNIX operating system. Here the addressing limits are practically non existant. In order to run the Unix system only the processor card and the software had to be purchased, 1 Mbytes of memory, disk controller card etc. being Q-bus compatible, were already available. A complete QU-68000 system will therefore cost about the same price as an LSI-11 system and is thus much cheaper than any other system so far used as a host for the M-68000 cross-software. After some premilinary difficulties due to the non standard Pascal compiler used on the PCS the following cross-software packages could be made available:

- M68MIL crossassembler
- Include processor (prepocessor allowing conditional compilation)
- Pascal cross compiler
- Modula-2 cross compiler
- Cufom linker (Cufom = Cern Universial Format for Object Modules)
- Cufom prepusher
- Cufom unipusher

Maintainance of the cross-software

All of this software was already used to generate a few test programs. It is however possible that, using more elaborate features, more bugs will show up. In order to ease debugging or implementation of new features into the cross-software packages, maintainance command files are available for each piece of software. The sources of the software packages combine code for different host machines. A preprocessor (include processor) allows conditional compilation using keywords (e.g include PCS ... will include code special to the PCS host machine). Therefore the preprocessor must be run for each software package, specifying the correct keywords, before it can be translated by the host's pascal compiler. This is done by a command file whose name consists of the letters 'pre' followed by the name of the software piece to be translated:

- preasm for the assembler
- prelink for the Cufom linker
- preprepush for the Cufom prepusher
- preunipush for the Cufom unipuisher
- prepass1 for pass1 of the Pascal compiler
- prepass2 for pass2 of the Pascal compiler
- premodula for pass1 of the Modula-2 compiler
  (pass2 is identical to pass2 of the Pascal compiler)

Once the source is available the PCS Pascal compiler, the linker etc. has to be run in order to get a program executable under UNIX. This is done using a so called 'makefile' (see UNIX manual for details). Again for each piece of cross-software such a makefile is available called 'make' followed by the name of the software piece to be treated.

- makeasm to build the M68MIL assembler
- makelink to build the Cufom linker
- makeprepush to build the Cufom prepusher
- makepass1 to build pass1 of the Pascal compiler
- makepass2 to build pass2 of the Pascal compiler
- makemodula to build pass1 of the modula compiler

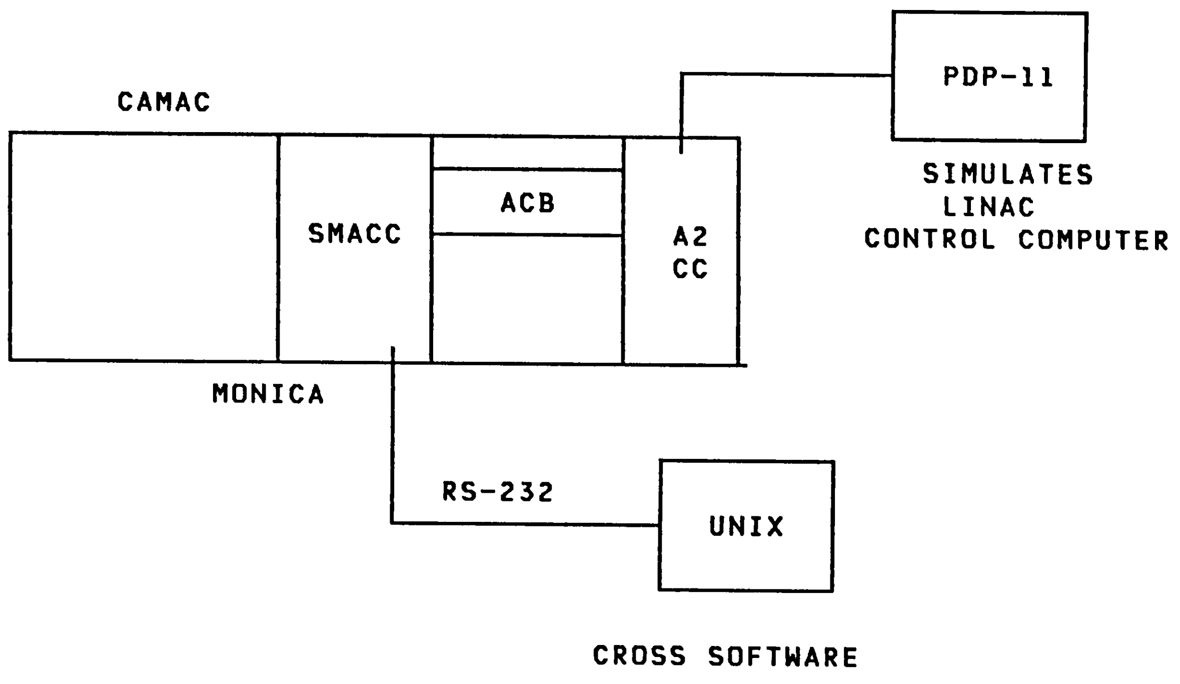

Figure 1: Setup for development of application programs

An application program is developed under UNIX on the PCS. For each of the different steps involved in order to get a working application program e.g. assembly, linkage, push, an interactive command file is available. These command files, so called UNIX shell scripts, ask the user for all information that is needed. The names of the shell scripts are:

casm: (Cross assemble) for the M68MIL assembler This shell script also allows subsequent linkage and push

cpas: (Pascal cross compiler) runs both passes of the Pascal cross compiler This shell script also allows subsequent linkage and push

cmod: (Modula-2 cross compiler) runs both passes of the Modula-2 cross compiler This shell script also allows subsequent linkage and push. In the current version a standalone runtime package is linked to the user program.

clink: Runs the Cufom linker

cpush: Runs the Cufom pusher

The final result (an S-record file) is loaded into the SMACC via an RS 232 line, using the loader facility of the monitor resident in the SMACC. An additional possibility is given by transfering the S-record file to a PDP-11 connected to CAMAC. A standard software package is used to do this (Kermit). A loader program, called LSM (LoadSMacc) reads the S-format file and loads it via CAMAC into the SMACC. This is particularly useful to load the systems debug monitor (containing the S-format loader for the RS 232 line). LSM is a Fortran 77 program running under RSX 11M.

3. Software available on the SMACC

The following software is available on the SMACC itself:

debug monitor:	The DD MoniCa debugging monitor allowing inspection of memory locations and user registers, setting of breakpoints tracing of a user program, loading of S-records via the second serial port, disassembler, line assembler, symbolic debugging etc..
Camac library:	A collection of subroutines conforming to the Esone standard and implementing the CERN calling conventions. These subroutines are therefore callable from high level languages.
FPI library:	A collection of subroutines allowing front panel interrupt handling with Pascal interrupt routines.
Mailbox Driver:	Reading and writing of a mailbox memory within the SMACC. These routines confirm to the specifications given by D. Kemp in his Linac Note 85-4.

4. Conclusions and Future plans

Conclusions

By using standard software packages available from DD, a powerful development system for M-68000 based controllers could be supplied within a relativly short time (4 man month). Even documentation is not needed since it is available as writeups or yellow reports from DD.

Future plans

A possibility of upgrading the system is to implement the RMS68K real time system, offering all support needed for real time applications. In order to do this, substantial changes in the MoniCa monitor are needed. Remote access to SMACCs using the serial CAMAC link is also envisaged. These upgrade will be done as soon as the need for these features arises. The main goal however is to get our first application program, the beam switching control, to work. This program will be entirely written in one of the high level languages available.

Acknowledgements

We are very grateful to D. Wiegandt DD, David Foster DD, Horst von Eicken DD from who we got the cross-software sources and who helped us to get the system up. They very patiently helped us when we were stuck with a problem.