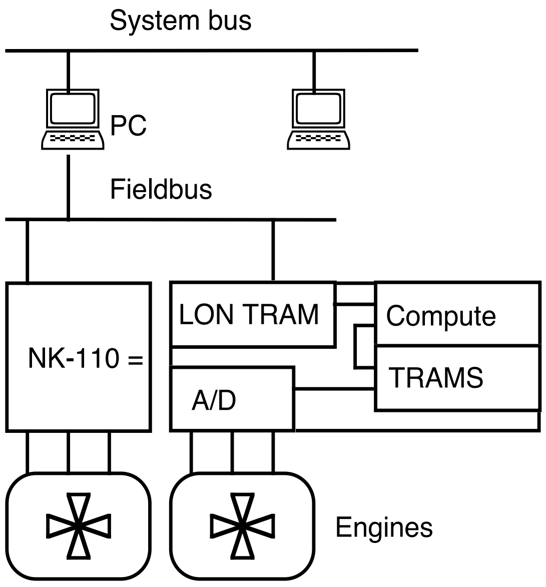


Nordic Transputer User Group
3rd Conference Copenhagen 14.15-May 1993

Transputers and MS-Windows - Case Study of a Development Process. Extended Summary

Øyvind Teig[1] and Thor Vollset[2]

1. The product: Autronica NK-100 MIP-Calculator

The main purpose of NK-100 is to calculate MIP and measure operational parameters of a diesel engine. MIP is the theoretical *Mean Indicated Pressure* that, if it were applied to the cylinder explosion chamber, delivers the same power in a diesel engine, as the actual pressure inside the cylinder does during normal use. In effect it gives a measure of the delivered power. This involves fast data-acquisition, real-time calculation and presentation. Curves and data are presented graphically to the engine operator. The system is used for maintenance and thus improves engine operation. Autronica has pioneered the development of MIP-Calculators for 20 years.

2. System architecture

The system bus is a network running the TCP/IP protocol. The host computer is a PC using the MS Windows[3] operating system. The PC is connected to the fieldbus through a B004 interface and a LON-TRAM. The fieldbus is a 2-wire multi-master bus called LON[4]. Inside the embedded controller NK-110 transputer modules perform data-acquisition and computation. Connection to the fieldbus is through an Autronica designed LON-TRAM. A single transputer link connects the A/D-hardware. Pressure transducers and a rotational angle transducer are connected to the engine.

2.1. Topology

Despite LON being a multi-master bus the NK-100 controllers are connected as a pipeline. There is no communication between the controllers; therefore the logical connection is a star with the PC in center.

[1] Autronica A/S, N-7005 Trondheim Norway Tel.: +47 7 581000 Fax.: +47 7 919320
[2] Tordivel, Waldemar Thranes gt.77 N-0175 Oslo Norway Tel.:+47 22206890 Fax.:+47 22206891
[3] Microsoft Windows
[4] LON = Local Operating Network from Echelon

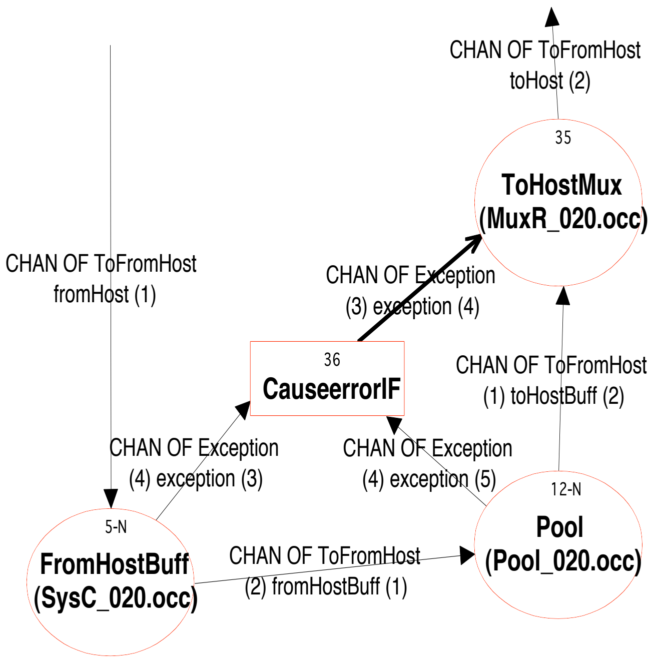
2.2. System bus and TCP/IP

The data exchange on the system bus uses the MITS[5] protocol. MITS is a Norwegian initiative to standardize data exchange in maritime environments, both protocol and format. Format through Companion Standards which defines all messages from an object and the structure of a message with its entities. The basic software for MITS has been developed by SINTEF Reguleringsteknikk in Trondheim, Norway. MITS was originally implemented in C running under UNIX. Autronica has ported the protocol to MS Windows using a Berkeley sockets TCP/IP-library called Distinct[6]. Mapping UNIX processes to MS Windows applications showed that a UNIX design easily maps to MS Windows despite its non-preemptive multitasking.

2.3. Communication between Windows and transputers.

The transputer tools often use the ISERVER protocol to communicate with a root transputer. Currently there is no support under MS Windows, therefore we made our own proprietary protocol implemented in Pascal using B004 functionality. It is implemented as a Dynamic Link Library, DLL, which means that any programming language can access this module to communicate with a network of transputers.

2.4. Details of how we defined PROTOCOLs



The transputer code is written in occam[7]. In occam PROTOCOLs must be defined in an include file when communicating between modules. We have used a convention with only one PROTOCOL per include file. A typical name for a protocol is "ToFromHost" where data flow in both directions are defined in the same PROTOCOL. This was done to directly map data-flow diagrams to the protocol; see figure. The Host communication is done "blocking" (master-slave) or "event-driven". Blocking means that the HOST does not send a new command before an answer to the previous blocking command has been received. Event-driven communication may be one of two types: (1) either the HOST program sends a message or (2) the transputer in NK-110 sends a spontaneous event, data or error messages, to the HOST program.

[5] Maritime Information Technology Standard, NTNF, Norway
[6] From Distinct Corporation
[7] occam-2 from Inmos

2.5. Fieldbus and LON

Autronica designed a size 2 LON-TRAM with one LON communication processor and two INMOS link adaptors. The links are defined to be "up" and "down" and are physically connected in a pipeline. The LON-TRAM is designed to support virtual channels to save communication bandwidth.

2.6. Data collection and computation

The compute TRAMs communicate with the data-acquisition hardware through a transputer link. This maps very well to the Occam programming model. The compute TRAM tells the hardware to start sampling and then the transputer reads the specified number of values from the input link. This is very efficient due to the DMA-mechanism in the transputer's link engine. The values are pressure and rotational data, and calculations are done after a complete data-set has been acquired. Computation and data-collection are parallel processes.

3. Tools and methodology

3.1. Programming languages

The programming languages Borland Pascal[8] and occam are used. Programming under MS Windows needs a powerful object-oriented language which Borland have in Pascal. Occam is the way to program an embedded transputer system. Using occam removes the need for a separate real-time kernel, and is a "clean" and secure language for implementation of any real-time system.

3.2. Data-flow diagrams and entity-relation diagrams

We have used ABC-Flowcharter[9] as a manual CASE-tool and have followed some guide-lines. We have made data-flow (see figure previous page) and entity-relationship diagrams. We found that data flow and entity relationship diagrams are "orthogonal". An entity relationship diagram show the objects of the application and the dataflow diagram the communication between these objects. ABC-Flowcharter supports hierarchical decomposition. By double-clicking on an object the next level is entered. The names in the diagrams were mapped directly to the code.

3.3. Text-editor for documentation and Help

The cut and paste mechanisms of MS Windows has made it easy to combine tools like ABC-Flowcharter and MS Word[10]. MS Word is used for documentation.

The help in NK-100 is based on the MS Windows hypertext help system. We used a macro-package for Word called RoboHelp[11] to simplify making the Help-system.

3.4. Program-editor for occam and Turbo-Pascal

We use Origami, a DOS-based folding text-editor made by Martin Green[12], for program development, both for Occam and Pascal. Before deciding to use Origami we, in cooperation with the author, added a number of missing features. The main enhancements were templates, cut-and-paste and complete case-sensitive find/replace/replace all.

---

[8] Borland Pascal for Windows
[9] From Roycore Inc.
[10] From Microsoft
[11] From Blue Sky Software.
[12] Martin Green, UK

Combining Origami with Borland Pascal IDE[13] is easy. The integrated Pascal editor has features like "chroma coding", jump to compile-error, integrated symbolic debuggers and context-sensitive help. These are features which are necessary for a productive programming environment. Origami adds its folding power. In Pascal the power of folding has become vital in showing the object structure and navigating between objects and units. On our wish list is the folding concept built into Borland IDE.

We experience two problems with folding editors, to be able to switch between a folded and flat view, and the temptation of not breaking the program down into its natural components with subroutines or functions. Cut and Copy fold are very/too powerful. The first problem is solved by alternatively using Origami and a non-folding editor such as the integrated editor, the second problem is only solved by pure discipline or by scrupulous author-reader rounds.

3.5. Coding conventions and rules

We have been using strict conventions in layout and names for the code production phase. This has been done to reduce and/or eliminate the need for comments, make the produced code look like pseudo-code and make a code reviewer able to spot logical errors. The general rules applied have been to avoid abbreviations, use descriptive names, naming constants and prohibiting one character names. The layout has been standardized through a set of templates for common structures. To achieve reuse of modules and procedures we have been looking for generic structures and methods. In addition we have followed this rule:

> **Procedures / functions shall, as a rule, be small and perform one action involving no more than two objects or be built using such procedures/functions.**

3.6. Building occam programs - some hints

When programming in occam we have been able to do it in "pure" occam - that is to say that all compiler switches have had the default value during all compilations. We have been thinking CSP[14]-wise, so that we have used PARs quite much. Whenever an #INCLUDE or #USE has been used referring to files that reside on the same directory, a list of filed folds of the same has always been included. This makes navigation between files quite fast and increases readability. Occam PROTOCOL was used extensively - nothing was put into a long array and decoded by hand so to say. CHAN OF ANY was not used. Variables were extensively abbreviated[15] into constants like this: "**VAL INT ThisValue IS thisValue:**" Observe the capital letter convention - only variables have small letters in the first letter of a name. We try to make declarations, functions and procedures as local as possible. This gives us a problem for PROCs: should every value used be parameterized? The solution here is that if the procedure seems to realize a generic function, we pass all values as parameters. Occam FUNCTIONs are very good in use, since they cannot have side-effects. Observe that this is very different from Pascal or C functions! A problem with occam is that it is not possible to make generic multiplexors when PROTOCOLs are used. Using virtual channel routing CHANs can use the same link, so that multiplexors / demultiplexors are eliminated.

4. Concluding remarks

We have described how the NK-100 product has been developed. It has been a major effort to implement an embedded transputer system controlled from a modern Graphical User Interface and connected to a system bus based on MITS and TCP/IP. Learning Windows programming and practicing transputer occam programming - and connecting the two together demanded a lot of effort. But at the end of the project we see an extremely powerful framework waiting for new challenges.

✉

[13] Integrated Development Environment

[14] Communicationg Sequential Processes - Tony Hoare's classical work from 78/85 - basis for occam and the real-time part of ADA.

[15] The **IS** clause is an element of an occam "abbreviation"