CHILL – the International Standard Language for Telecommunications Programming

BY KRISTEN REKDAL

681.3.04

Abstract

Computer controlled telecommunication systems have proven to contain some of the largest and most complex pieces of software ever constructed. To master this complexity requires the use of powerful methods and tools. The CCITT High Level programming Language – CHILL – is an enabling technology that has contributed to making such systems possible. This paper surveys the status and history of CHILL and gives a short introduction to the language. The Norwegian and Nordic activities supporting the CCITT work are described. Finally the development of the CHIPSY toolset for CHILL programming is summarised.

1 Software is important in telecom

With the increasing use of software in telecommunication systems, programming has become an important enabling technology for telecommunication services. The proportion of software cost over hardware cost in the development and production of telecom systems has been shifting rapidly. In 1970 the software cost was practically 0 % of the total, in 1980 it was around 50 %, while presently it has risen to around 85 %. More and more the services earning money for the operating companies are implemented in software.

This shift towards software in telecom was realised by some pioneers already in the late 1960's when the CCITT made initial investigations into the impact of computerised telephone exchanges. However, even when work on the CHILL language started in 1975, nobody dared to predict the present extent and pervasiveness of software in telecom systems.

Most of the basic technical problems to be faced in telecom programming were already at that time fairly well understood, and programming languages had been developed or were being developed to cope with these problems. One may therefore ask why CCITT should engage in making yet another programming language, adding to the already abundant flora of such languages. There were three major reasons:

- to remedy weaknesses of existing languages when applied to telecom systems, because such systems have special needs
- to consolidate, in one language, features only found in a variety of other languages
- to provide one standard language covering all producers and users of computerised telecom systems.

The standardisation aspect was motivated by the fact that the telecommunications community was a very large one and distributed all over the world. Most producers and operators of such equipment would somehow come into contact with software. Thus, the benefits of standardisation also in the field of software were likely to be substantial.

2 Telecom software is special

Experience has clearly taught us that there are many features of telecom software which differentiate it from other types of software. Typically telecom software exhibits some or all of the following characteristics:

- Real-time, concurrent behaviour, time critical response
- Very large size, e.g. 100,000 – 10,000,000 lines of source code
- High complexity
- High performance requirements
- High quality requirements, e.g. reliability, availability, fault tolerance
- Use of multiple, heterogeneous, networked computers, e.g. host and target computers
- Long lifetime, e.g. 3 – 20 years
- Continuous development during the lifetime, spanning several technology generations in both software and hardware
- Large development teams, the same personnel rarely stays with the system for its lifetime
- No single person has full understanding of the whole system
- Geographic separation of development activities
- Delivered in many copies in several variants and generations
- Total lifetime cost much higher than initial development cost
- Strong influence from international standards, e.g. CCITT, ISO.

CHILL was designed to meet the challenges of such software. In retrospect it was clearly justified to develop a special language for telecom programming.

3 CHILL is a viable language

Since its inception in 1975, CHILL has grown to become a major programming language within telecom. There are now at least 12,000 – 15,000 CHILL programmers in the world. More than 1,000 man-years have been invested in CHILL compilers, support tools and training. Many of the most successful telecom switching systems on the world market have been engineered in CHILL. See Table 1.

Two other ways of illustrating the extent of CHILL usage are shown in the graphs below. Figure 1 shows that six of the top ten world telecom equipment manufacturers are using CHILL for their major switching products. The manufacturers are shown according to turnover in 1989 for telecom products. The major CHILL users are shown in black columns. Non-CHILL users are shown in white columns. Partial users are shown in shaded columns.

Figure 2 shows which programming languages are actually most used in the telecom industry for public switching systems. The volume measure is the percentage of the total number of installed public digital local lines per year, for the years 1985 to 1990. The figure for 1990 is an estimate. The graph shows that the usage of CHILL has been steadily increasing. 45 % of the digital local lines installed in 1990 were supported by software written in CHILL. This is up from 0 % in 1980. CHILL is the only programming language common to more than one of the major public telecom switching systems. The CHILL column is the sum of the installed lines for EWSD, E10, D70, and System 12.

The second most used telecom programming language is Protel used by Northern Telecom, while C, used by AT&T, is in third place.

Table 1 Telecommunication switching systems programmed in CHILL

System name	System type	Manufacturer	Country
EWSD	Public exchange	Siemens	Germany
PRXD	Public exchange	Philips	Netherlands
System 12	Public exchange	Alcatel	Belgium/Germany
E10	Public exchange	Alcatel	France
D70	Public exchange	NTT, NEC, Hitachi, Fujitsu, Oki	Japan
KBD70	Public exchange	Oki Electric	Japan
LINEA UT	Public exchange	Italtel	Italy
RN64	Cross connect	Telettra	Italy
	PABX	PKI	Germany
TDX-10	Public exchange	Daewoo Telecom	Korea
Tropico	Public exchange	Telebras	Brazil
PXAJ-500/2000	Public exchange	10th Research Institute	China
Levent	Rural exchange	Teletas	Turkey
Nodal Switch	PABX	Alcatel Telecom	Norway
HICOM	PABX	Siemens	Germany
Saturn	PABX	Siemens	USA
Amanda	PABX	Alcatel	Austria
Sopho	PABX	Philips	Netherlands
Focus	PABX	Fujitsu	Japan
TTCF	Teletex	Ascom Hasler	Switzerland

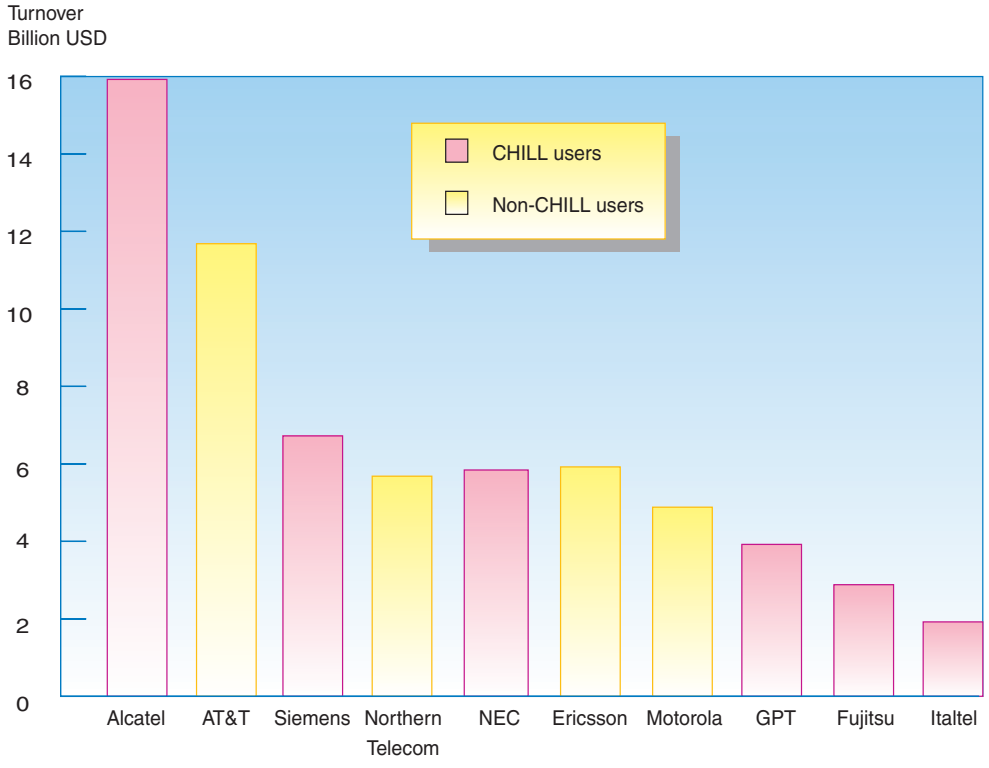

Figure 1 CHILL usage among major telecom manufacturers

4 CHILL development started in 1975

The groundwork for the development of a high-level telecom programming language started already in 1966. The CCITT plenary assembly in 1968 in Mar del Plata, Argentina, decided that such development should be handled by the CCITT. The necessity of standardising a language of this type for switching functions was recognised, as was the need for a unified language for both delivery and technical specifications.

In the study period between 1968 and 1972 the CCITT mainly concerned itself with the definition of the material to which the standards were to apply. Recommendations for software specifications were looked for, as were unified descriptions of software-controlled systems (or SPC – Stored Program Control – systems, as they were called at the time) so that the different telephone switching systems could more easily be compared with each other.

The new language to be developed was required to be easy to read and learn while at the same time being open-ended and system-independent with regard to technical innovations.

The outcome of the discussions was that it was deemed necessary to standardise not only one, but three types of language:

- a *specification and description language,* later to become SDL, for defining descriptions of features, specifications and system designs
- a *high-level programming language,* later to become CHILL, for actual coding, and also
- a *man-machine command language,* later to become MML, for operation and maintenance by the user of SPC switching systems.

In the 1973-1976 Study Period, specific work was started to create this family of languages for SPC use, CHILL, SDL and MML. SDL stands for the CCITT Specification and Description Language (2). MML stands for the CCITT Man-Machine Language (3).

As the responsibility of CCITT was confined to public (telephone) switching systems at the time, it is important to note that the scope of the CCITT standards, including the languages, was confined to that application domain.

The work of CCITT in 1973 started with an investigation and evaluation of 27 existing languages. From this set a short-list of six languages was made. They were:

- DPL, made by NTT, Japan
- ESPL1, made by ITT (now Alcatel), USA and Belgium
- Mary, made by SINTEF/RUNIT, Norway
- PAPE, made by France Telecom/ CNET
- PLEX, made by Ericsson, Sweden
- RTL2, made by the University of Essex, UK.

The conclusion of this study was, however, that none of the languages were satisfactory for the intended application area. In 1975 an ad hoc group of eight people called "The Team of Specialists" was formed to handle the development of a new language. The team had representatives from

- Philips, Netherlands
- NTT, Japan
- Nordic telecom administrations
- Siemens, Germany
- Ellemtel, Sweden
- ITT (now Alcatel), USA
- British Telecom
- Swiss PTT.

A preliminary proposal for a new language was ready in 1976. The language was named *CHILL* – the CCITT High Level Language.

In the following Study Period, starting in 1977, it was decided that there was a need for an evaluation of this proposal by practical experience in order to complete the language. For this purpose the Team of Specialists was replaced by "The Implementors Forum". Its task was to encourage and gather experience from trial implementations of CHILL. By the end of 1979 a language proposal was completed. The CCITT Plenary Assembly approved the CHILL definition in November 1980 to become CCITT Recommendation Z.200 (1).

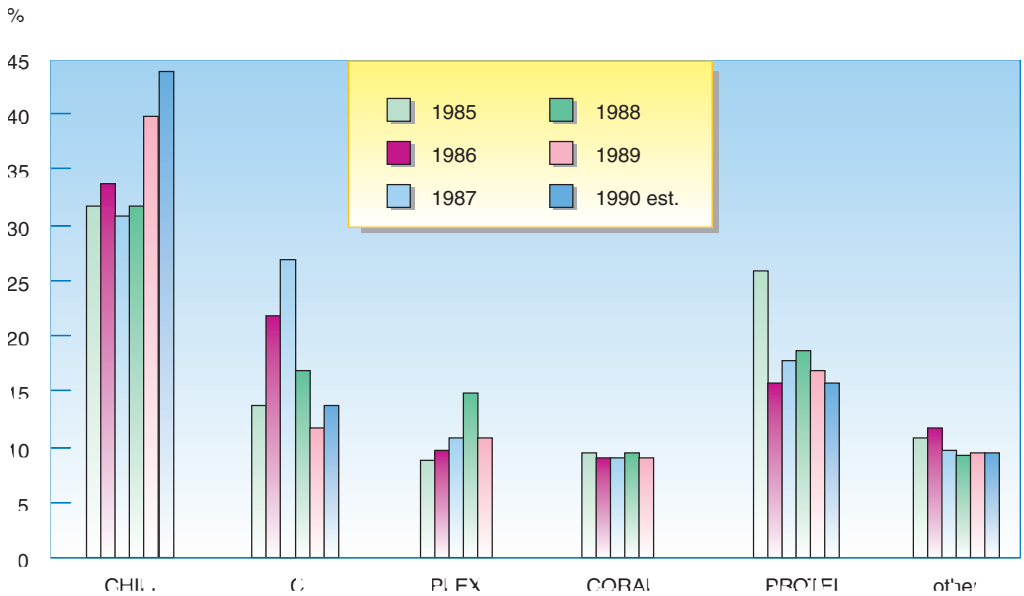

Figure 2 CHILL usage in terms of installed digital local lines per year

CHILL inherited most of its traits form other high-level programming languages, drawing upon the best of the language developments of the 1970's and 1980's. CHILL became a truly state-of-the-art language. Box one below shows the essential groups of constructs in CHILL while box two shows a program fragment giving a flavour of the appearance of the language.

From 1977 many companies and organisations took up the challenge of constructing CHILL compilers. Until 1987 more than 30 compilers had been made for many different computers. Several of the compilers were put to use in industrial product development. Most of the systems listed in Table 1 were programmed by means of compilers developed in-house.

Since 1980 the CCITT has continued to support and maintain CHILL and upgrade the language according to needs from the industry and new technical developments. The main extensions to the language have been:

1981-84	Piecewise programming, name qualification, input/ output
1985-88	Time values and timing operations
1991-92	Data objects for real numbers
1993-96	Work on object orientation is planned.

In 1989 CHILL was recognised as ISO/IEC Standard 9496.

5 Norwegian Telecom was actively involved

Norwegian Telecom had the foresight to enter into the CCITT language work at an early point in time. When the CCITT had decided, in late 1974, that a new language had to be developed, Norwegian Telecom Research (NTR) took an interest to participate actively. NTR managed to enlist the telecom administrations of Denmark, Finland, Norway, and Sweden in a concerted Nordic effort to sponsor an expert from the SINTEF research institute to participate in the CHILL Team of Specialists.

This Nordic representative became the vice chairman of the group from 1976 to 1980 and the chairman of CHILL development from 1980 to 1984. Thus the Nordic effort in this development was highly visible.

In addition it was decided to establish a Nordic support group covering the activities of all the three CCITT languages. This group had representatives from the four countries mentioned above plus Iceland. Later also British Telecom joined. The group was very active and met several times a year from 1975 to 1984.

6 CHIPSY – the CHILL Integrated Programming System

One specific outcome initiated by the Nordic co-operation on CHILL has been *CHIPSY* – the CHILL Integrated Pro-

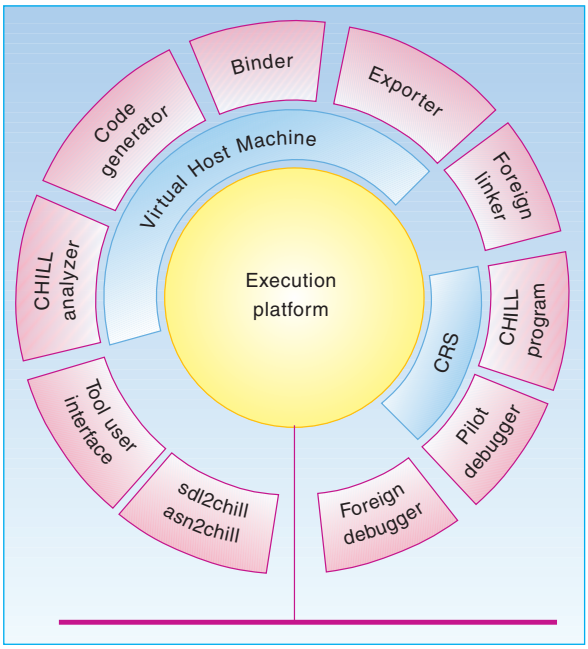

Figure 3 The CHIPSY concept

Box one – Sample CHILL program

```
line_allocator:
MODULE
    SEIZE line_process, line, occupied, unoccupied,
        search,connect, accepted;
    GRANT line_allocator_process;

    line_allocator_process:
    PROCESS ();
        NEWMODE states = SET(free, busy);
        DCL next_state states := free, lno INT := 0;
        line(lno) := START line_process(lno);
        DO FOR EVER;
            CASE next_state OF
            (free): RECEIVE CASE SET sender;
                (occupied):
                    next_state := busy;
                (search):
                    SEND connect(sender) TO line(lno);
                    SEND accepted TO sender;
                    next_state := busy;
                (else): -- Consume any other signal
                ESAC;
            (busy): RECEIVE CASE SET sender;
                (unoccupied):
                    next_state := free;
                (search):
                    SEND rejected TO sender;
                (else): -- Consume any other signal
                ESAC;
            ESAC;
        OD;
    END
    line_allocator_process;
END
line_allocator;
```

gramming System. CHIPSY is an open-ended support environment for programming in CHILL. CHIPSY is oriented towards the development and maintenance of real-time software, especially in the area of telecommunications.

CHIPSY is a family of productivity tools for telecom software development. The toolset comprises compilation tools, associated real-time operating systems and source level test and debug tools. See Figure 3.

The CHIPSY tools cover the design (partly), programming, execution and testing phases of the software development cycle.

Where convenient and useful, CHIPSY interfaces to products or components produced by other companies, e.g. the text editor Emacs (7) or the SDL tool SDT made by TeleLOGIC Malmö AB (6). CHIPSY has been in regular use for the development of major industrial telecom software since 1980, comprising several million lines of CHILL source code. CHIPSY's reliability has been proven by the fact that there are now in operation several hundred digital switching units world wide programmed with CHIPSY. Several units are operating in high load, mission critical (e.g. military) applications.

6.1 Development of CHIPSY since 1977

The CHIPSY development started in early 1977 when the telecom administrations of Denmark, Finland, Norway and Sweden decided to sponsor a CHILL compiler implementation project at SINTEF. Later also British Telecom (now BT) joined in funding the project.

All the development work was carried out by SINTEF until 1984. Then CHIPSY was taken out of SINTEF to establish a new company, later called KVATRO A/S, for the purpose of commercial exploitation of the research results.

The major milestones in the history of CHIPSY have been:

1974 The Mary language was shortlisted by CCITT as one of six candidate languages for a CCITT telecom programming language. Mary is a machine oriented high level programming language developed by SINTEF from 1971 to 1973.

1975 SINTEF is sponsored by the Nordic administrations to join the CCITT Team of Specialists to design the CHILL language.

1977 Start of CHIPSY compiler development project hosted on the ND-100 16-bit minicomputer, initially targeted to the Ericsson APZ210 switching processor, later changed to Intel 8086.

1980 First industrial contract. CHIPSY licenses were sold to Alcatel Telecom Norway for use in developing a digital military communication system.

1982 CHIPSY was sold to Ascom Hasler AG. Hasler signs a contract with SINTEF to implement a CHILL level debugger called CHILLscope.

1983 Code generators implemented for the bare Intel 80286 microprocessor and the ND-100 computer with the SINTRAN III operating system. A new, portable CRS – CHIPSY Real-time Operating System – written in CHILL for both the 80286 and the ND-100/SINTRAN III.

1984 CHIPSY sold to 10th Research Institute of the Ministry of Posts and Telecommunications of China. A new company, later known as KVATRO A/S, was founded in order to commercialise CHIPSY. KVATRO was given the rights to market and further develop CHIPSY by the Nordic telecom administrations.

1985 Rehosting of CHIPSY cross compilers to VAX/VMS was completed.

1978 CHIPSY native compiler and new generation CRS ready for VAX/VMS.

1990 Completion of rehosting and retargeting to several UNIX platforms. CHIPSY becomes the first CHILL compiler available on a 386 PC platform. CHIPSY licenses sold to Nippon Telegraph and Telephone Software Laboratory.

1991 Extending CHIPSY to cover distributed processing. CHIPSY becomes the first CHILL compiler available on a laptop computer.

1992 Implementation of an ASN.1 to CHILL translator completed. Implementation of an SDL to CHILL translator completed. First implementation of the Pilot debugger for testing and debugging in distributed real-time systems.

1993 Full implementation of the Pilot debugger. CHIPSY is ported to SPARC workstations.

6.2 CHIPSY is the outcome of a co-operative effort

Until 1993 more than 100 million NOK (15 million USD) have been invested in the development and maintenance of CHIPSY. The investments constitute co-operative efforts with various partners. The main contributors to this development have been:

- Telecom administrations of Denmark, Finland, Norway, Sweden and United Kingdom acting in concert
- Telecom Norway
- Alcatel Telecom Norway
- Ascom Hasler
- SINTEF
- 10th Research Institute of MPT, China
- Norwegian Industrial Fund
- Nippon Telegraph and Telephone
- KVATRO A/S.

7 Conclusions

CHILL is unique because it is a *standardised* programming language catering for the needs of the telecommunications community. It has already gained a solid acceptance within the industry world wide, and has been used for the construction of some of the world's most successful switching systems.

Because of the longevity of telecom systems and the persistence of programming languages, CHILL will continue to have a strong impact in the world of telecommunications well beyond the year 2000.

It is safe to say that CHILL has largely achieved its original objective of becoming a standard language for the programming of public telecom switching systems.

Box two – CHILL Overview

CHILL is a strongly typed, block structured language. A CHILL program essentially consists of data objects, actions performed upon the data objects and description of program structure.

Data modes

The categories of data modes (data types) provided in CHILL are discrete modes, real modes, powerset modes, reference modes, composite modes, procedure modes, instance modes, synchronisation modes, input-output modes and timing modes. Some modes may have run-time defined parameters. In addition to the CHILL standard modes, a user may define other modes.

Data objects

The data objects of CHILL are values and locations (variables). Locations may be created by declaration or by built-in storage allocators. The properties of locations and values are language defined to a detailed level and subject to extensive consistency checking in a given context.

Sequential actions

Actions constitute the algorithmic part of a CHILL program. To control the sequential action flow, CHILL provides if action, case action, do action, exit action, goto action, and cause action. Expressions are formed from operators and operands. The assignment action stores a value into one or more locations. Procedure call, result and return actions are provided.

Concurrent execution

CHILL allows for concurrent execution of program units (processes). Creation and termination of processes are controlled by the start and stop actions. Multiple processes may execute concurrently. Events, buffers and signals are provided for synchronisation and communication. To control the concurrent action flow, CHILL provides the start, stop, delay, continue, send, delay case and receive case actions, and receive and start expressions.

Exception handling

During run-time the violation of a dynamic condition causes an exception. When an exception occurs the control is transferred to an associated user-defined handler.

Time supervision

Time supervision facilities of CHILL provide means to react to the elapsed time of the external world.

Input and output

The input and output facilities of CHILL provide the means to communicate with a variety of devices of the outside world.

Program structure

The program structuring statements are the begin-end block, module, procedure, process and region. These statements provide the means for controlling lifetime of locations and visibility of names. Modules are provided to restrict visibility in order to protect names against unauthorised usage. Processes and regions provide the means by which a structure of concurrent executions can be achieved. A complete CHILL program is a list of modules or regions that is surrounded by a (imaginary outermost) process.

Piecewise programming

This facility allows large programs to be broken down into smaller, separate handling units.

CHILL has also spread beyond public switching and has been used for a number of other telecom products like PABXs, non-voice equipment, etc.

CHILL has stood the test of time. Even though CHILL is now almost 20 years old, it has not been outdated. On the contrary it is flexible enough to fit into new technological contexts. For example, experience has shown that:

- CHILL is well suited as a target language for translations from SDL (5) and ASN.1 (8)
- CHILL concurrency concepts are well suited for implementing distributed systems (4).

CHILL has proven to be powerful enough for implementing modern telecom systems which are much larger and more sophisticated than could have been foreseen in the 1970's, e.g. ISDN, IN and mobile communication systems.

Finally, the CHILL work has provided a foundation for commercial product developments.

References

1 ITU. *CCITT High Level Language (CHILL).* Geneva, 1980-92. (Recommendation Z.200.)

2 ITU. *Functional Specification and Description Language (SDL).* Geneva, 1976-92. (Recommendation Z.100.)

3 ITU. *Man-Machine Language (MML).* Geneva, 1976-92. (Recommendation Z.300.)

4 *CHIPSY Reference Manual.* Trondheim, KVATRO A/S, 1993.

5 Botnevik, H. Developing Telecom Software with SDL and CHILL. *Telecommunications,* 25(9), 126-132, 139, 1991.

6 *SDT Reference Manual.* Malmö, TeleLOGIC Malmö, June 1992.

7 *GNU Emacs Manual.* Free Software Foundation, 1986.

8 ITU. *Specification of Abstract Syntax Notation One (ASN.1).* Geneva, 1988. (Recommendation X.208.)