VI. DLT'S IMPLEMENTATION: HARDWARE AND SOFTWARE.

1. Storage capacity.

1.1. Storage requirements.

An MT system generally requires a large amount of storage capacity. Apart from a transient workspace during the translation process, permanent storage of the system's software - including grammars and dictionaries - must be provided for.
DLT is no exception in this regard. Its projected storage requirements are even considerably higher than those of most other MT systems, because of the planned extensive inclusion of collocations, microcontextual procedures, etc. in the DLT dictionaries [see III.3.2.5, III.3.3.3, III.4.2.4e]. Loosely, one could say that the DLT translation method derives its strength from an increased use of lexicon-based information, in addition to the fact that the 'double-translation' principle [see III.1.4. and III.2.3] requires a double quantity of lexicon.

A detailed design of the format of lexicon entries and an assessment of their exact space requirements based on DLT's needs in connection with lexicographic and statistical properties of the various languages is outside the framework of this feasibility study. The same applies to other components of the DLT software, such as grammars. For all these, we have to work with careful estimates, partly based on comparison with other MT systems, partly on intuition. As will be shown in the hardware design below, there are no critical storage capacity limits at stake, which permits an estimation approach. Only one storage estimate has been made with a comfortable physical size in mind (the estimate for the size of a lexicon entry).

A general remark concerns compaction. Although some of the storage figures presented here may appear high in comparison with other MT systems, rigorous compaction has been assumed consistently, implying of course the existence of conversion mechanisms between readable (external) and bit-compressed (internal) formats for the sake of development and maintenance. This is exemplified by the deliberately designed compaction of the morphem-coded IL [see IV.5 and also III.5.2], whose variable-length bit-patterns will be used throughout the lexicons, including many collocations and microcontextual procedures stated in terms of IL [see III.4.3].
For SL- and TL-elements, variable-length character-coding may be used, among other techniques, to compress lexicons, indexes etc. [see also Martin, 1977: 517,581].
As a consequence, the byte as a unit of storage will have

less importance in detailed memory and formatting considerations for DLT; nevertheless, it is used in the gross storage calculations here because of ease and convention.

The projected extension of DLT lexicons with encyclopedic 'world knowledge', to enable the use of AI during translation, still further reinforces the demands for storage space and compaction. Again, the latter is assured by the dominant use of compacted IL [see also III.6.2].

Table VI-1 gives an overview of the most important storage needs (a more comprehensive overview is given in section 1.2). We distinguish between 'fixed' and 'variable' information:

'Fixed' information is part of the DLT system or terminal equipment sold to the user (who will call it 'system software'). This software will be stored within the user's terminal permanently, to be activated at the beginning of each translation session. Periodic updating and maintenance of the 'fixed' software (especially the lexicons!) will be necessary.

'Variable' information consists of the user's text passing through the system (the 'data'), and all its intermediate representations, derived trees etc. during the translation process. It also includes the lexicon extract (a collection of lexicon entries) temporarily made available for the processing of a particular piece of text (basically a sentence). 'Variable' information is overwritten or removed many times during and at the end of a session.
An exception is the saving of a statistics block. For a selected circle of users, DLT will record the disambiguation dialogues and related information, as part of the system's long-term optimization and development. Cumulated statistics blocks will then have to be transferred to the DLT support center periodically.

Table VI-1 has further been divided according to whether the calculated space is required in the SL-module only, in the TL-module only, or in both. We will come back upon the separation of SL- and TL-modules and their corresponding system software parts later in this chapter. We proceed with explanations on some separate items of the table now:

<u>The size of a lexicon entry.</u>
A fixed size of 512 bytes (4096 bits) is projected here. In comparison, the size required for entries of a consultative electronic dictionary has been calculated at 2800 bits (36% of which goes for speech output) [Fox,1980]. Such a consultative dictionary contains 65 English words per entry on average, for definitions, examples of use, synonyms. Though an MT dictionary serves a different purpose, its needs show some parallels: semantic subclassification, valency patterns and microcontextual

'FIXED' INFO:		
SL- or TL-	Lexicon-entry size: 512 B (10.000 entries: 5 MB) (30.000 entries: 15 MB) 50.000 entries:	25 MB
SL-	Lexical SL-parser (incl. spelling correction):	25 KB
	Syntactical SL-parser (SL-grammar):	60 KB
	Paraphrase generator (for disambiguation dialogues):	50 KB
	IL-recognizer (error-filtering):	40 KB
TL-	IL-parser:	25 KB
	TL-synthesis (incl. IL-TL transfer):	50-150 KB
'VARIABLE' INFO:		
SL- or TL-	Gross lexicon extract (incl. 'buckets'):	200 KB
SL-	SL-parser output ('trails', worst case):	30 KB
	Statistics (average session):	25 KB

Table VI-1. Overview of the most important storage usage in DLT.

procedures take the place of definitions and examples, while synonyms will be included in DLT dictionary entries too (for paraphrase generation in disambiguation dialogues). Taking into account the more compressed (primarily machine-readable) and - at the same time - more exhaustive nature of the DLT lexicon (including collocations, idioms, etc.), an estimate of more than twice the size of the consultative lexicon (minus speech output) appears safe.
Note that the fixed size of 512 bytes applies to SL-IL and to IL-TL entries separately, i.e. 1KB reckoned over an SL-TL pair, which is nearly 3 times as high as the figure quoted for the Russian-to-French GETA ARIANE-78 system [Whitelock, 1982].
Also note that extralingual AI-related additions, the so-called 'world knowledge', is NOT included in the estimated size of the lexicon entry. The large needs of this encyclopedic information - which will probably be organized in a different way than the lexicon proper - are taken into account separately in this chapter.

SL-grammar.
Of course, the compact object version is understood here, not the PROLOG source. The figure of 60 KB is based on the reported size [Bates, 1978: 238] of a large semantic ATN-grammar: 448 states, 881 arcs and 2280 actions. Assuming 4 bytes/state, 12 bytes/arc, 20 bytes/action and 50 memberlists of 40 bytes each, we arrive at approx. 60 KB. Semantic grammars are generally much larger than syntactic grammars; on the other hand, DLT may require more elaborate augmentations.
The estimates for the IL-parser and IL-recognizer have been made been made intuitively and in relation to the SL-grammar.

Gross lexicon extract.
During the processing of a sentence, all its related lexicon entries (10 on average, see Table VI-2) must be kept available. For the fast retrieval of one lexicon entry - here more properly called 'record' - , a bucket of 20-40 records will be involved (based on an estimated index size of 1250-2500 keys to access a lexicon of 50.000 records).
In fact, the surrounding bucket contents can be disposed of as soon as the retrieval of the record (e.g. by binary search of the bucket) has been completed. If a search fails, the bucket is useful in connection with spelling-correction (minimum-distance match). But depending on the available storage space, garbage removal may not at all be attractive (except at incidental storage demand peaks). For this reason we assume a permanent 200 KB occupation, the contents of which change per sentence.

SL-parser output.
As a worst case, a momentary cumulation of 60 parse trails [see III.4.2.3 for the concept 'trail'] during the processing

of a sentence has been assumed. A parse trail may be expected to require up to 10 times as much information as the final BCE representation of a sentence (400 bits), i.e. 4000 bits.

Statistics.
We reckon with an SL-translation session of maximally 120 sentences, i.e. 2 hours at a speed of 1 sentence per minute (including typing of the source text). Per SL-sentence, a disambiguation dialogue of 3 question-answer cycles is taken as an average [see Table VI-2]. To record the course and the outcome of this dialogue (including generated paraphrases, microcontext, etc.), a space demand of 4 times the space of a BCE sentence is guessed at.

The above considerations all concern quantity requirements, but of course there are other aspects as well. The total set of storage requirements for DLT can be characterized as follows:

- abundant quantity
- sufficient speed
- desk-top proportions
- software distribution facilities
- low costs

In a system which relies on human input and interactive assistance for one part, and medium-speed data transmission to end-user terminals for another, memory access speed is not as critical as it would be for a high-throughput batch MT system. Nevertheless, speed conditions have been checked here in the section on timing relations [VI.2], in conjunction with the storage realization proposed below.

Reminding the reader to the 'outside look' of DLT [Chapter II], any storage solution must fulfil the demand for desk-top proportions. This includes economic power consumption as well as physical dimensions. No special provisions should be required to save the system software when power goes off.

In the 'distribution' concept of DLT, the translation process is decentralized, distributed over sending and receiving terminals. Each terminal translates with its own copy of the DLT system software for the one or the other language. As a practical consequence, one will want an easy way of distributing the software, exchanging it for new releases, arranging updates etc.

All these requirements must be brought into agreement with

each other and with the low-cost requirement, especially for the receiving (IL-to-TL) equipment: what we talk about is - eventually - the environment of videotex-enhanced TV's and personal computers, at home and on everybody's desk.

1.2. Storage realization.

The hardware configuration proposed in this chapter features 3 levels of memory, which will be discussed and detailed now in the light of the earlier-mentioned storage requirements.

Background Mass Storage.

As non-volatile high-capacity memory, we propose the optical disc, a videodisc-like product developed by Philips and also announced by several other manufacturers for computer application now [Sebestyen, 1982; Megadoc, 1983; Fujitani, 1983]. We will refer to this product by its Philips acronym: DOR (Digital Optical Recording). Though higher capacities are expected in the course of the 1980s, we will assume here a capacity of 1 GB (gigabyte), a conservative assumption for this new medium.

Fig. VI-1 gives an impression of how the large DLT lexicons and translation software easily fit on a DOR disc, leaving an enormous extension capacity to accomodate dictionaries for special terminology and AI-related encyclopedic knowledge.
The DOR disc is a write-once medium. Track-sectors of 512 bytes (Philips) are the units of reading/writing. A sector cannot be physically overwritten. In order to operate the disc as an updatable storage medium, we consider its logical capacity to be one fourth of its physical capacity, allowing all information to be updated four times on average (of course, software provisions will be needed for this). The only DLT category for which no updating requirement exists is the cumulated statistics block, which is in fact archival information (over 2000 translation sessions, 50 MB will have been cumulated, leaving 950 MB physical or 237 MB logical storage).

As part of a DLT terminal, the DOR will be used as follows: at the beginning (power-on) of a session, the translation software (200-400 KB for an SL-, 100-200 KB for a TL-terminal) will be transferred to the (volatile) on-board memory [see below]. As for the lexicon, separate entries (which coincide with track-sectors of the disc) will be accessed on request by the lexical parser when scanning the SL-input. What actually happens is the transfer of a 'bucket' of 20-40 lexicon entries, which will then be

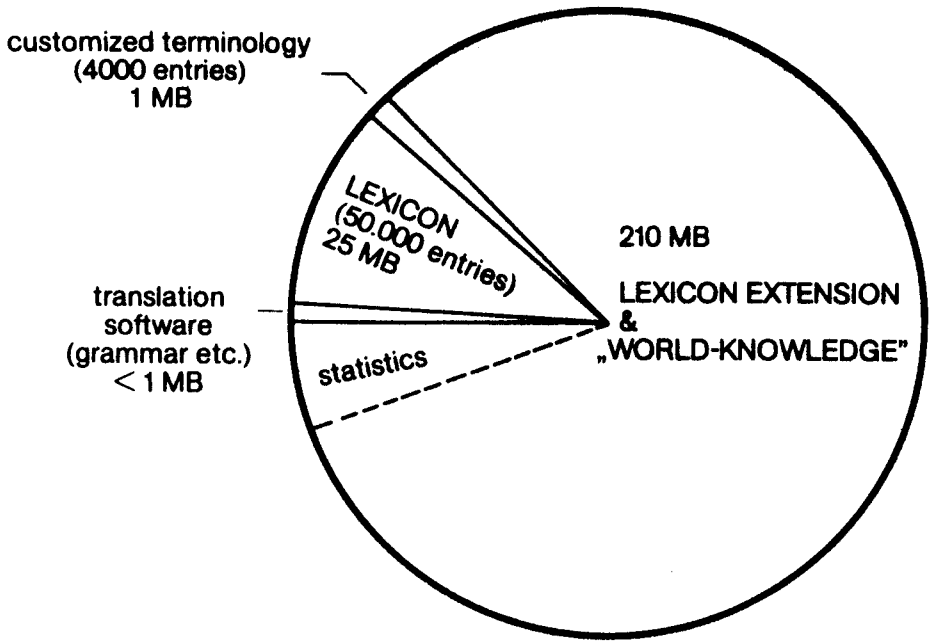


Fig. VI-1a. DOR containing a complete set of software for a 'one-way' DLT terminal, i.e. either an SL- or a TL-module (translating sender or translating receiver).

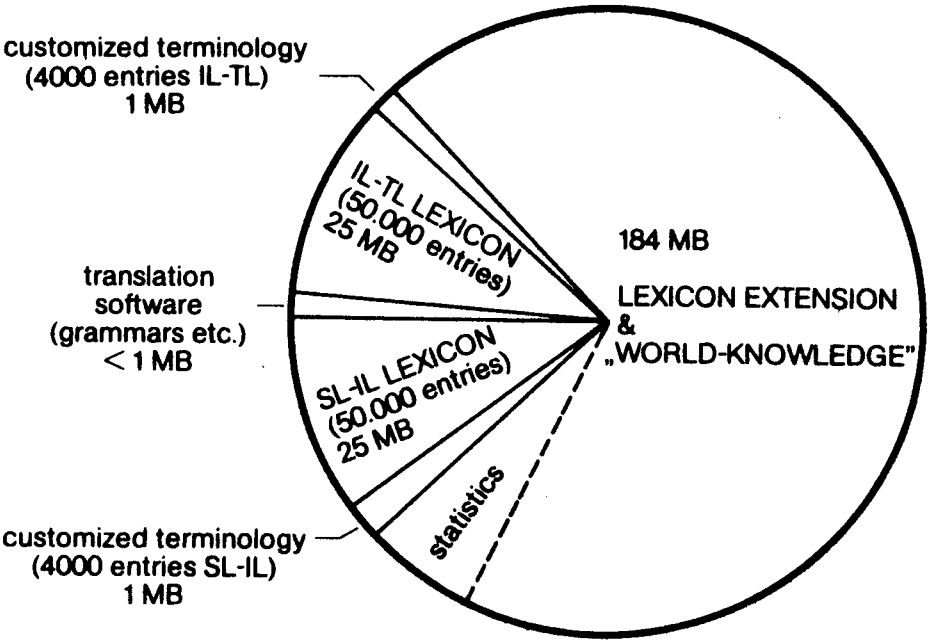


Fig. VI-1b. DOR containing a complete set of software for a 'two-way' DLT terminal, i.e. equipped with both an SL- and a TL-module (SL=TL). The world knowledge is written in IL and needs to be stored only once.

further searched in common main storage [see below]; this arrangement keeps indexing within proportions and also facilitates spelling correction.
The mean random access time for Philips DOR is 137.5 msec [Megadoc, 1983]. In our timing calculations [...], we have used a figure of 150 msec, which includes the transfer time for a bucket (the transfer rate for DOR is several Mbits per second). By and large, the access speed seems sufficient.
Reloading of software will occur if one exchanges the disc (e.g. if a German user leaves his terminal to a French user), or after a power-break. At the end of a session, a statistics block built in main storage is written to the disc.

DOR offers lowest cost-per-bit and the drives designed for it are reported [Fujitani, 1983] to consume less than 200 Watt, which makes them attractive for the office and home computer environment. Initial costs announced for this product ($ 6000 - $ 7000 in OEM quantities, early 1984) still are prohibitive for its inclusion in a 'consumer product', but expectation of a steady price decline through the 1980's seems justified, especially if combined drives for data processing as well as audio/video appear on the market [Sebestyen, 1982].

The fact that the DOR disc is removable satisfies the requirement for easy software distribution. Updates can also effectuated by down-line loading of user terminals from a DLT support center, as long as the transmission time and costs do not become prohibitive.

Integration of the DOR into a computer system is facilitated by the availability of controllers which are de facto standards for 5.25 and 8 inch Winchester disc drives. These controllers perform error correction, which is vital for reliable operation of the compactly coded software of DLT. The residual error after correction, quoted as 1 in 10**12 bits, corresponds to 1 in 10.000 translation sessions, assuming a transfer of 100 Mbits (including AI) per session.

Though the DOR excellently meets the DLT requirements, Winchester appears acceptable as a second choice, considering its increased compactness and cartridge removability. Especially for the development period in the next few years, as long as DLT's lexicons are still limited and the DOR is still expensive, a small Winchester (e.g. 18 MB) will be attractive. Apart from that, its mean access time (40 msec) is better than that of DOR.

Other alternatives that presented themselves have been rejected: dynamic RAM for the whole DLT lexicon would

require bulky battery back-up provisions and remain more expensive, without solving the software distribution problem. As for MBM, there is no consensus about their future support in industry.

Common Main Memory.

This is a volatile memory for the fast handling of transient data during the translation process. It will be realized by a 1 MB dynamic RAM board, connected to the disc controller and processors by the configuration's main bus [see section 3]. The various process-steps of the DLT translation (to be realized by separate processors, as will be explained in this chapter) have commonly access to this main workspace. Apart from information of the DLT category 'variable' [see Table VI-1], the common storage board will also accomodate lexicon extracts.

As with the background mass storage, one must consider separate main memory boards for SL- and TL-modules, because these modules will be distributed over separate terminals (the very essence of DLT!). We therefore have the following typical use of common main memory at send and receive side:

SL-side:	- SL-IL lexicon entries of current sentence (incl. 'buckets'):	200 KB
	- current IL sentence (BCE):	400 bits
	- macrocontext (up to 120 sentences), SL and IL:	18 KB
	- SL-parser workspace ('trails'):	30 KB
	- dialogue statistics:	25 KB
	- extract of AI-related world-knowledge ('scenario'):	100 KB
	- cumulated SL-parser results in case of postponed-dialogue mode [see section 2]:	500 KB
TL-side:	- IL-TL lexicon entries of current sentence (incl. 'buckets'):	200 KB
	- current IL sentence (BCE):	400 bits
	- macrocontext (up to 120 sentences), IL and TL:	18 KB

As appears from this list, a 1-MB capacity is sufficient at the SL-side, and abundant at the TL-side. When SL-modules would require higher-capacity boards in the future, these will be available off-the-shelf (as 2-MB boards already are).

On-board Memory.

Of the 3 levels of storage for DLT, this one provides the fastest access. As for the common main memory, dynamic RAM is used, but the on-board memory is faster because a processor can access it without having to contend for the computer's main bus. On-board memory is therefore particularly suited to hold instructions.
So whereas the common main memory will primarily be used for 'data' (including DLT lexicon extracts), the on-board memory is primarily intended for 'programs' (parsers, algorithms) and will be loaded with them at the beginning of each translation session (system booting).

Fig. VI-5 shows the amounts of on-board RAM and a brief indication of their main contents for the various processors that make up the DLT terminal, the sending as well as the receiving terminal [the motivation for parallel processors is given in sections 2 and 3]. Evidently, each processor has a specified task, for which the programs and tables reside on the same board. The following list sums up the proposed storage capacity and its estimated utilization for each processor board (SBC = single-board computer):

SL-side: SBC 1 - Lexical Parser (128 KB):
- lexical SL-parser (incl. algorithms and tables for spelling correction): 25 KB
- index to the lexicon (1250 index entries of 8 bytes): 10 KB
- word-syntax analyzer (for compounds not in the lexicon): 15 KB

SBC 2 - Syntactical Parser (256 KB):
- syntactical SL-parser (complete SL-grammar in ATN-form): 60 KB
- dictionary of 200 function words (pronouns, prepositions, etc.): 10 KB
- high-frequency dictionary of 240 other words: 120 KB

SBC 3 - AI Processor (512 KB):
- advanced semantics and macrocontext-oriented AI algorithms: 200-400 KB

SBC 4 - Dialogue Processor (256 KB):
- dialogue-disambiguation planner (best sequencing of questions): 30 KB
- paraphrase-generating software (SL-synthesis): 50 KB

SBC 5 - IL-coder (128 KB):
- tree-to-string conversion (ordering, adding of agreements, insertion of separating elements): 20 KB
- IL-recognizer (comprehensive IL-grammar in ATN-form, with error-filtering): 40 KB

TL-side: SBC 6 - IL-decoder & TL-synthesis (256 KB):
- IL-parser (IL-grammar in ATN-form): 25 KB
- IL-TL transfer & TL-synthesis software: 50-150 KB

The list indicates ample dimensioning of the on-board memories. For SBC 3, the AI-board, extension of the on-board memory (currently limited to 256 KB) with a piggy-back or separate memory board may be necessary (this can be connected onto the AI-board's local bus).

The three-level memory facilities discussed above provide the best solution for the total set of DLT storage requirements. With regard to capacity, there are practically no restrictions: the development of AI-related software will very probably always lag behind the expanding storage sizes offered by the hardware technology.

2. Timing relations.

In order to check the sufficiency of storage access speed and to assess the required degree of parallel processing, the timing relations in DLT must be considered. These involve human as well as computer process components of the semi-automatic DLT translation process. The human activities (typing, thinking, reading) are generally slow and time-consuming, compared to the performance of computer equipment. With the hardware configuration proposed in this chapter, one would expect the following order of (potential) speed bottle-necks:

- human speed (typing: 30 sec/sentence,
  thinking: 30 sec/dialogue,
  reading: 2-5 sec/sentence);
- mass storage access speed
  (DOR: 1.5 sec/sentence);
- network transmission speed
  (1200 bps: 0.3 sec/sentence);
- main storage access speed
  (incl. contention for main bus);
- processor speed

The art of the DLT hardware design exists in exploiting the fact that the human speed is the real bottle-neck. This again emphasizes the character of DLT as a non-batch translation system.
Figs. VI-2 and VI-3 show the timing relations for a whole translation cycle, based on model assumptions listed in table VI-2. A more detailed timing diagram of a part of the translation proces is given in fig. VI-4. For sake of clarity, all these diagrams use average values; in practice, statistical variation of typing, dialogue and processing time will of course occur.

Fig. VI-2a shows one complete translation cycle, starting with typing of the SL-sentence, followed by the interactive disambiguation dialogue, and ending with the reading of its TL-translation. As DLT is a distributed system, SL-text generation and disambiguation (in the 'sending' terminal) and display of the TL-translation (in the 'receiving' terminal) are - in principle - separated in space and time, a gap to be bridged by transmission and intermediate storage. In the diagrams of figs. VI-2 and VI-3, this main interface 'gap' (which would be somewhere in the middle of Tr2-6) has not been

accounted for. In fact, fig. VI-2a represents a non-standard DLT mode of usage: the translation of each sentence is read by the operator before he types the next one. This practice will only be observed during system development, for demonstrations and for special applications in which bilingual operators require immediate feedback [see V.4]. For development and for verification purposes, a loop-test (in which TL=SL) can be useful. Besides, the time profile of fig. VI-2a can represent some other human activity (e.g. reading in the SL-manuscript) preceding the typing of the next sentence.

The standard-mode of DLT operation is represented by fig. VI-2b. The overlap of typing with the after-dialogue translation (Tr2-6) of the previous sentence underlines the 'decoupling' of the two 'halves' of the translation process. The parallelity of typing (and its simultaneous pre-translation Tr1) with Tr2-6 is however motivated by another reason:
If the operator at the SL-side is ready to continue typing immediately after the disambiguation dialogue, a delay of several seconds may be annoying. From dialogue design and response-time psychology considerations [Martin, 1973] we conclude that this delay will be less acceptable after a relatively short dialogue than after a relatively long one. As the figs. VI-2b through VI-2d suggest, the long-term evolution of DLT will yield a gradually shorter dialogue (balanced by an increasing AI component): whereas in VI-2b the average dialogue consists of three question-response pairs, in VI-2c this has been reduced to two and in VI-2d even to one. At the same time, the diagrams show an increase of Tr2-6, representing the increase in sophistication of TL-modules (translation quality improvement by macrocontext-oriented TL-word choice procedures [see III.6]).

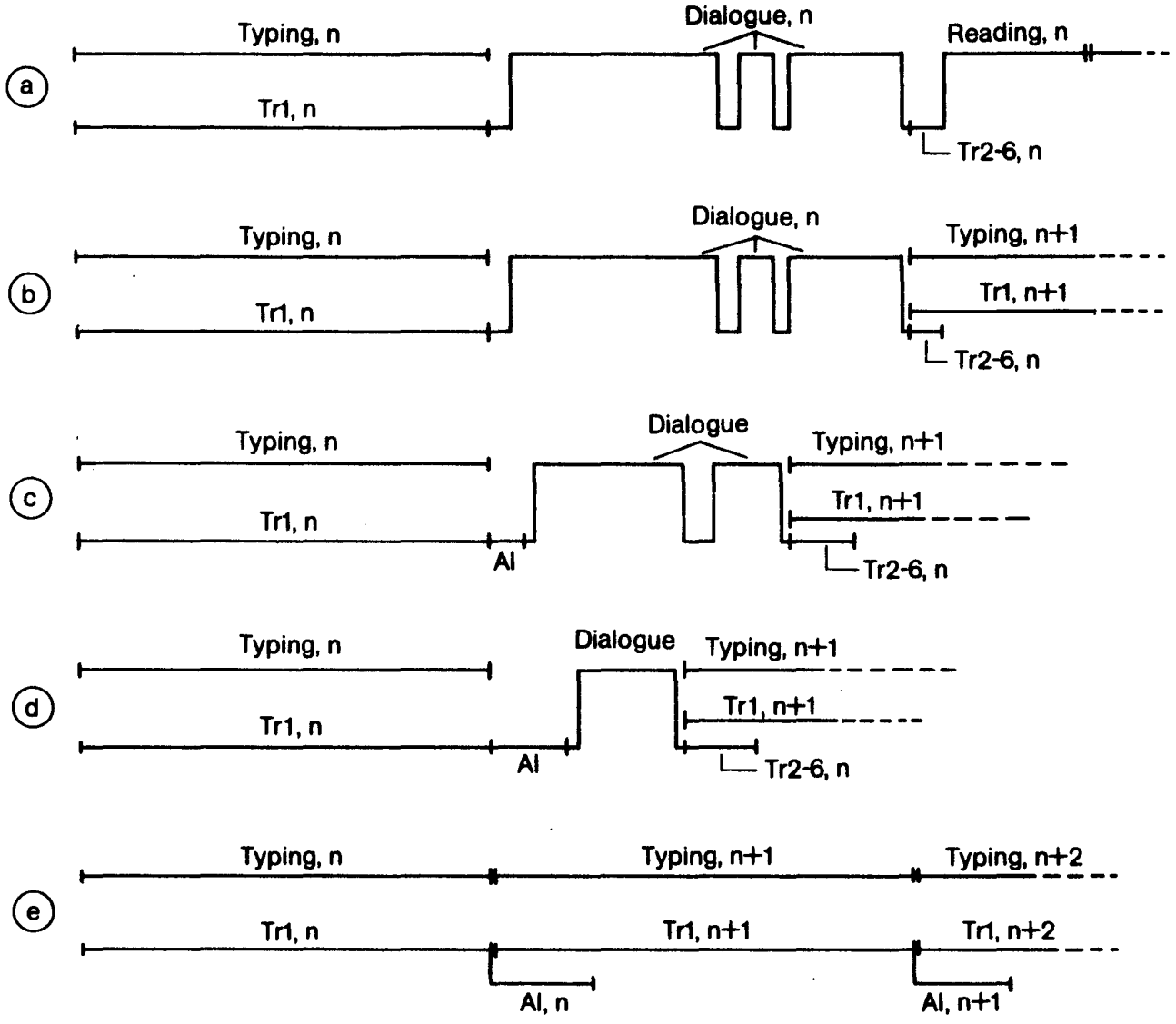
A potential delay could also develop before the dialogue: a gradually increasing future AI component might result in a substantial response time there. If that would be the case, the 'postponed-dialogue mode' (fig. VI-2e) should be used. All disambiguation dialogues (and consequently all subsequent process steps) are postponed till some time after the typing (and automatic pre-translation) of the text. This has the advantage that typing can go on uninterruptedly, with parallel automatic processing permitting the immediate completion of both the Tr1 and the AI parts. The postponed dialogues and further processing are shown in fig. VI-3g.
Apart from its future use for avoiding AI-induced delays, it should be remarked that the postponed-dialogue mode may well be preferred by operators who want to type their SL-text without being disturbed or distracted by any computer-initiated interaction (e.g. authors who create their text during typing).

In all the above cases, translation or at least the first part of it (Tr1) takes place immediately during typing. We refer to this as the 'immediate-translation mode'.

A different situation is covered by fig. VI-3: here the initial input to DLT is a file of SL-text already in the computer (previously typed or obtained by other means). We call this the 'queued-translation mode'. Notice that raw SL-text is assumed as input here (with the exception of fig. VI-3g, which belongs to the postponed-dialogue mode). The translation cycle is not affected by human typing speed now, and Tr1 can be compressed drastically.

With the disambiguation dialogue as the only human component left (except for the reading activity in fig. VI-3a), the aim is now to make the dialogue throughput as high as possible. This can be done by reducing the delay between dialogues, i.e. by parallelity between Tr2-6 and Tr1 (fig. VI-3c), but also by reducing the dialogue length itself, via a gradually increasing weight on the AI-component (figs. VI-3d and VI-3e). But similarly as in the immediate-translation mode of fig. VI-2, this again tends to increase the gap between successive dialogues, which justifies the arrangement of parallelity between Tr1 and the dialogue of the preceding cycle (fig. VI-3f). In fact, further compression is imaginable: Tr1,n+1 can already start during AI,n; such an overlap of (Tr1,AI),n+1 with (AI,D),n would reduce the inter-dialogue gap to about 1 sec, quite attractive with a typical dialogue length of 5-8 seconds. Accordingly, the translation throughput (at the SL-side, with parallel processors for Tr1, AI and D) would correspond to 6-9 sec/sentence,
The postponed-dialogue mode of fig. VI-3g (and fig. VI-2e) seems to give the same throughput without the need for parallel operation of AI and D. Its disadvantage however is the big amount of intermediate storage required to buffer the cumulated results of pretranslation.
For simplicity, we assumed here that the translation of sentence n+1 is independent of the translation (especially the disambiguation) of sentence n, which is certainly not true in general. This even questions the feasibility of fig. VI-2e (AI may not make much sense without full disambiguation of the preceding context). Also, it would not permit total compression of fig. VI-3f, limiting the throughput to 10 sec/sentence on average.

Like the postponed-dialogue mode, the queued-translation mode can be attractive for operators who prefer to type their SL-text uninterruptedly, and work through the dialogues afterwards. As DLT's AI-component will develop in the future, the queued mode will become even more attractive, because annoying AI-induced delays could build up otherwise.

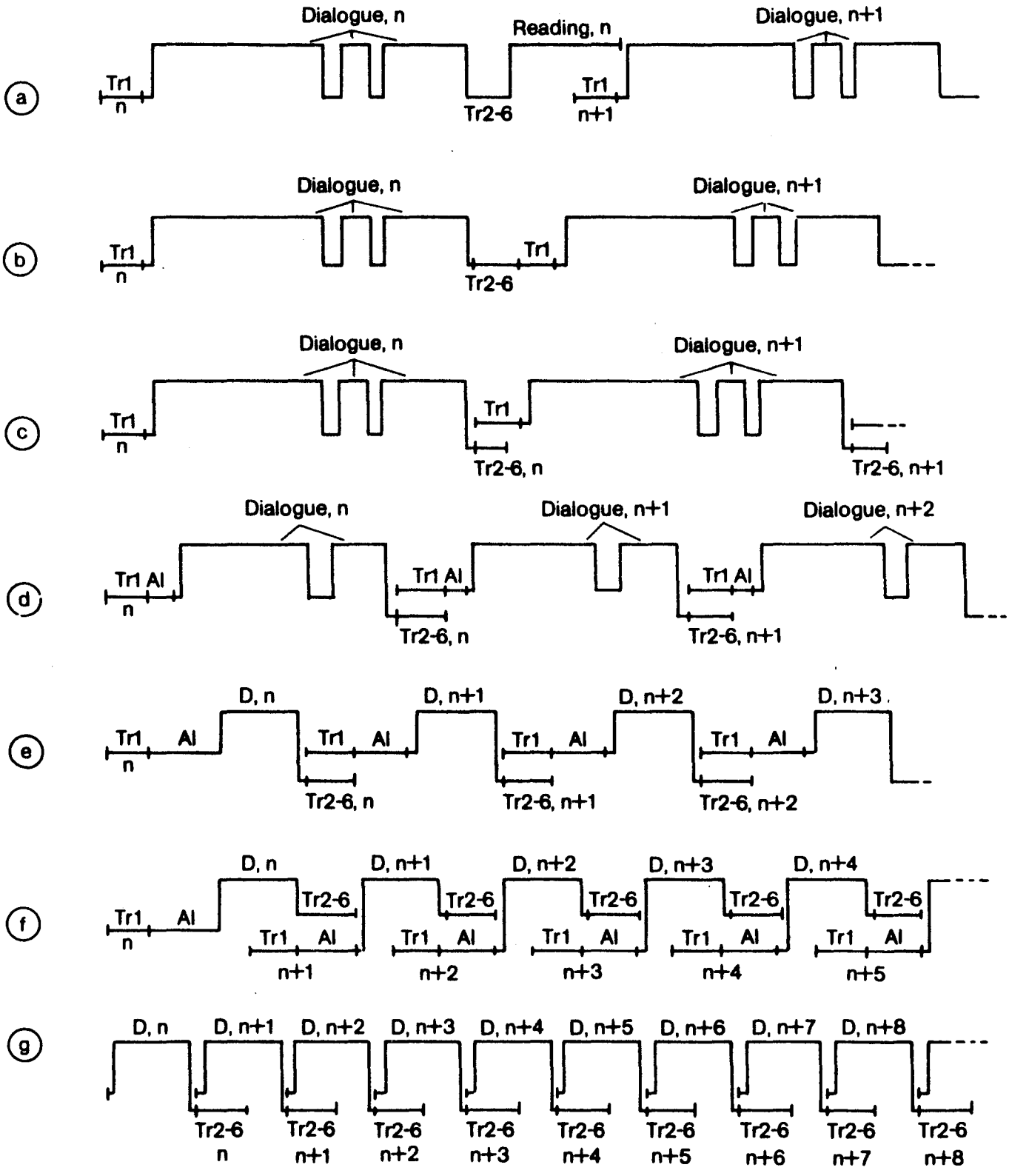


HOR. SCALE: 2 mm ≙ 1 sec

LEGEND:

Typing	= entering of SL-sentence from keyboard
Dialogue	= disambiguation dialogue
Reading	= reading of the translation (TL ≠ SL) or back-translation (TL = SL)
Tr1	= Step 1 of the translation process (except for AI and dialogue)
Tr2-6	= Steps 2 thru 6 of the translation process [see III. 4.1]
AI	= sentence-final advanced semantic part of Step 1 („Artificial Intelligence")
n, n+1,...	denote subsequent units of translation (sentences)

Fig. VI-2. Timing relations for DLT's 'immediate-translation' mode, shown for 5 stages (a through e) of future system evolution. Notice the parallelity of typing and processing (Tr1) in all stages.



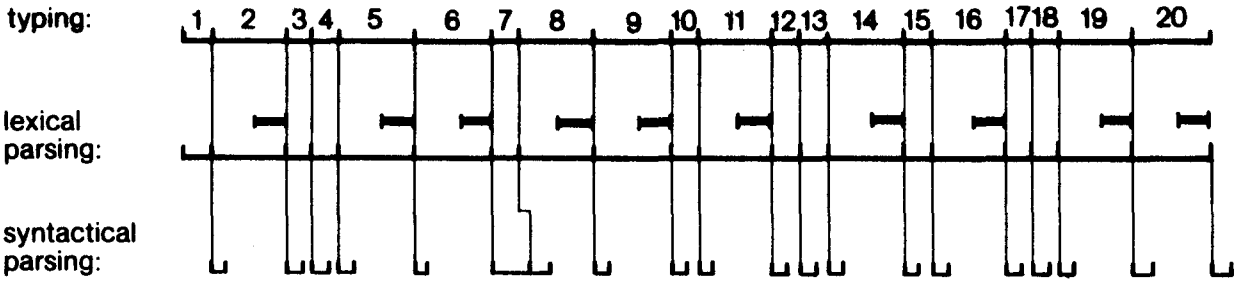
LEGEND: D = dialogue
[see also fig. VI-2]

HOR. SCALE: 2 mm ≙ 1 sec

Fig. VI-3. Timing relations for DLT's 'queued-translation' mode, again shown for various evolutionary stages. Note the gradual compression of dialogue activity.

The future extension of DLT with AI has deliberately been included in the timing considerations here, in order to derive a hardware design configuration into which it can be integrated easily and in a modular way, without disrupting the system's overall work rhythm and throughput. Such a configuration should therefore have parallel processors for Tr1, AI and D (D=dialogue), in addition to separate processors for Tr2-6.

A closer look on Tr1 makes apparent its two-level structure (fig. VI-4), corresponding with a lexical and a syntactical parser [see III.4.1]. Assuming a sentence of the model of Table VI-2, short function words alternate with longer content words. The lexical parser will follow the typing rhythm. Its task includes the retrieval of lexicon-entries from mass storage. The gross access time (incl. bucket transfer and binary search) for each content word has been separately indicated in the diagram.
The syntactical parser can only proceed when it receives another word from the lexical parser. As soon as this happens, syntactical processing - on the basis of the newly received word - is continued at full speed, independent of the typing or lexical parse speed. Mostly, the syntactical parser will be ready and idle before receiving the next word, but now and then (notably with 2- or 1-letter words, as the English 'I'), it may be overtaken by a new word's arrival.



LEGEND: HOR. SCALE: 5 mm ≙ 1 sec.

accessing lexicon entries from mass-storage

1, 2, 3..... denote the words of the SL-sentence

Fig. VI-4. Close-up of the simultaneous typing and Tr1 processing of a 20-word model sentence, showing the two parse levels. Despite its amount of idle time, the syntactical parser may occasionally lag behind (here at word 8).

human typing speed:	
4 chars/sec [cfr. Martin, 1973] (short bursts up to 8 chars/sec);	
language statistics:	
6 chars/word (incl. space or punctuation);	
20 words/sentence, of which:	
10 content words (typing time: 2 sec),	
10 function words (typing time: 0.7 sec);	
disambiguation dialogue:	
3 question-response pairs (with 'think' times of 3, 16 and 8 sec respectively)	
computer time Tr1 per sentence:	
10 DOR mass storage accesses (one for each content word, and 150 msec each [see VI.1]):	1.5 sec
syntactical parse time, estimated at the tenfold of the reported fast version of the 'LUNAR' ATN [Bates, 1978], which parses 8-12 word sentences in 150 msec:	1.5 sec
interleaved basic semantics, estimate:	0.5 sec
computer time Tr2-6 per sentence:	
10 DOR mass storage accesses (as above):	1.5 sec
IL-parse, transfer & synthesis, estimated at the fourfold of the 'LUNAR' ATN:	0.6 sec
microcontext-based word selection routines, estimate:	0.5 sec
IL transmission rate:	
3 sentences/sec (based on BCE: 400 bits/sec [see IV.5], and 1200 bits/sec transmission speed).	

Table VI-2. Model assumptions used in the timing diagrams of this chapter. As for the parallelity of the computer time components, storage access time will partially overlap with processing time.

Of the various technical solutions to handle this 'receive-overrun', considering also the clear functional separation between both parsers, the most attractive approach is to run them on two separate processors.

Summarizing our timing considerations, which have mainly been directed at the SL-side of DLT, we have the following throughput:

1 sentence/minute (incl. typing and disambiguation);
6 sentences/minute (without typing, with AI-support).

Of course, these figures are no more than global indications of averages: for a more complete and detailed picture (in which for instance the speed difference between a typing and a non-typing disambiguator is taken into account), the results of dialogue simulation [see Chapter VII, W.P. IV] must be awaited for.

At the TL-side, the translation proceeds fully automatically and is displayed at the terminal or TV screen, sentence after sentence. With a human reading speed of 2-5 sec/sentence, the processing time derived on the basis of table VI-2 (approx. 2 seconds) appears to be just enough. Accordingly, the throughput at the TL-side will be 30 sentences/minute.

In our timing diagrams, Tr2-6 has been used to indicate all translation steps after dialogue disambiguation. This implies the need for at least two processors, one for Steps 2, 3 and 4 (at the SL-side), the other for Steps 5 and 6 (at the TL-side). This brings the total number of processors to six.

On some of the six processors, the processing time consumed by grammars and algorithms will approach or even exceed the mass storage access time per sentence (see Table VI-2; note that the interleaved basic semantics and the microcontext-based word selection routines run on the same processors as the parsers). This means that DOR speed is not so much a critical factor in the DLT design.

Of course, the various time estimates made in the framework of this feasibility study can only provide an indication of timing relations. For a complete, detailed and reliable assessment a trial system based on the proposed hardware will be needed. As for the future extension of DLT with AI- and macrocontext-based procedures, the number of storage accesses (encyclopedic knowledge) as well as the processing time (algorithms) may increase.

3. Hardware implementation.

Three types of implementation can be considered for a DLT terminal:

- a 32-bit minicomputer
- specially developed VLSI
- current technology 16/32 bit micros

A 32-bit minicomputer must be rejected: it collides with the desk-top environment (compact terminals) envisaged for DLT, and would make the unit price of a commercial product too high. However, a 32-bit mini configuration can be recommended for development of software and dictionaries [section 4.2].

Specially developed VLSI is extremely expensive and could only be contemplated in a more advanced stage of DLT development, when confidence in selling very large production quantities will have been acquired.
Secondly, current technology microprocessors already represent the VLSI state-of-the-art, and are readily available.
Thirdly, the type of device being developed for the 'fifth generation' machines [Moto-oka, 1982; Treleaven, 1982], which would be close to the type required for DLT, will at least partially rely on replicated general-purpose processors.

This leaves current technology 16/32 bit processor chips as the most attractive basis for DLT hardware development.

The timing relations [section 1.2] of the DLT translation process (and its distribution over an SL-side and a TL-side terminal) showed the need for 6 processors: 5 for the SL-module and 1 for the TL-module. The programs to be run by each processor have already been listed in section 1.2.

At least for the hardware prototype, off-the-shelf Single-Board Computers (SBC's) with an adequate amount of on-board dynamic RAM and the usual interface provisions are proposed. These SBC's communicate with each other via a main bus [fig. VI-5], to which also a common 1 MB RAM storage board is connected.
The 'peripheral' functions of the DLT terminal will be dealt with by separate 8-bit processors on the SBC's and by additional standard boards: a communications controller, a disk controller (for the DOR), and a video board.
The SBC's will also have 32 KB PROM each (for booting, diagnostic programs, etc.).

As for the choice of microprocessor, the application requires:

- a cycle time in the nanoseconds range;
- simple addressing structure for several MB of storage (32-bit addressing);
- easy manipulation of addresses for fast tree searching;
- built-in string manipulation and string matching functions.

The processor currently on the market which best meets these requirements, is the Motorola MC68000 (also made by Mostek and Philips/Signetics). Of course, we do not preclude the use of newer processors, not yet announced.
The proposed bus architecture is the VME bus of Motorola, Mostek and Philips/Signetics. This bus satisfies the technical requirements for speed (MHz range) and addressing (up to 4 GB). It is compatible with the Eurocard standard, and the required racks and cards for such a system are readily available from several (European) manufacturers. A 'bus arbiter' [not shown in fig. VI-5] is required as a separate card to support the VME bus system.

A prototype for such a Eurocard-based DLT system [fig. VI-5] will require the following number of boards (incl. 3 peripheral boards and the bus arbiter):

- 10 boards for a separate one-way SL-terminal;
- 6 boards for a separate one-way TL-terminal;
- 12 boards for a combined two-way terminal,

which could fit in (e.g. a 15-slot) 19-inch rack, possibly in a tabletop chassis. The hardware costs, based on 1983 retail prices (in Dutch guilders, excluding VAT), are estimated at:

- Dfl. 124.000 for the one-way SL-terminal;
- Dfl. 58.000 for the one-way TL-terminal;
- Dfl. 151.000 for the two-way terminal,

including a small Winchester (18-34 MB). The attachment of a DOR unit will probably require Dfl. 25.000 more (the choice of DOR or Winchester for mass storage has been motivated in section 1.2.). In addition, 6 man-months must be accounted for basic software support, including the testing and validation of inter-processor communication.

The use of a Eurocard-based prototype will provide maximum flexibility for experimentation. It will also permit gradual development, in such a way that the pilot project proposed in Chapter VII could make use of it from the completion of the

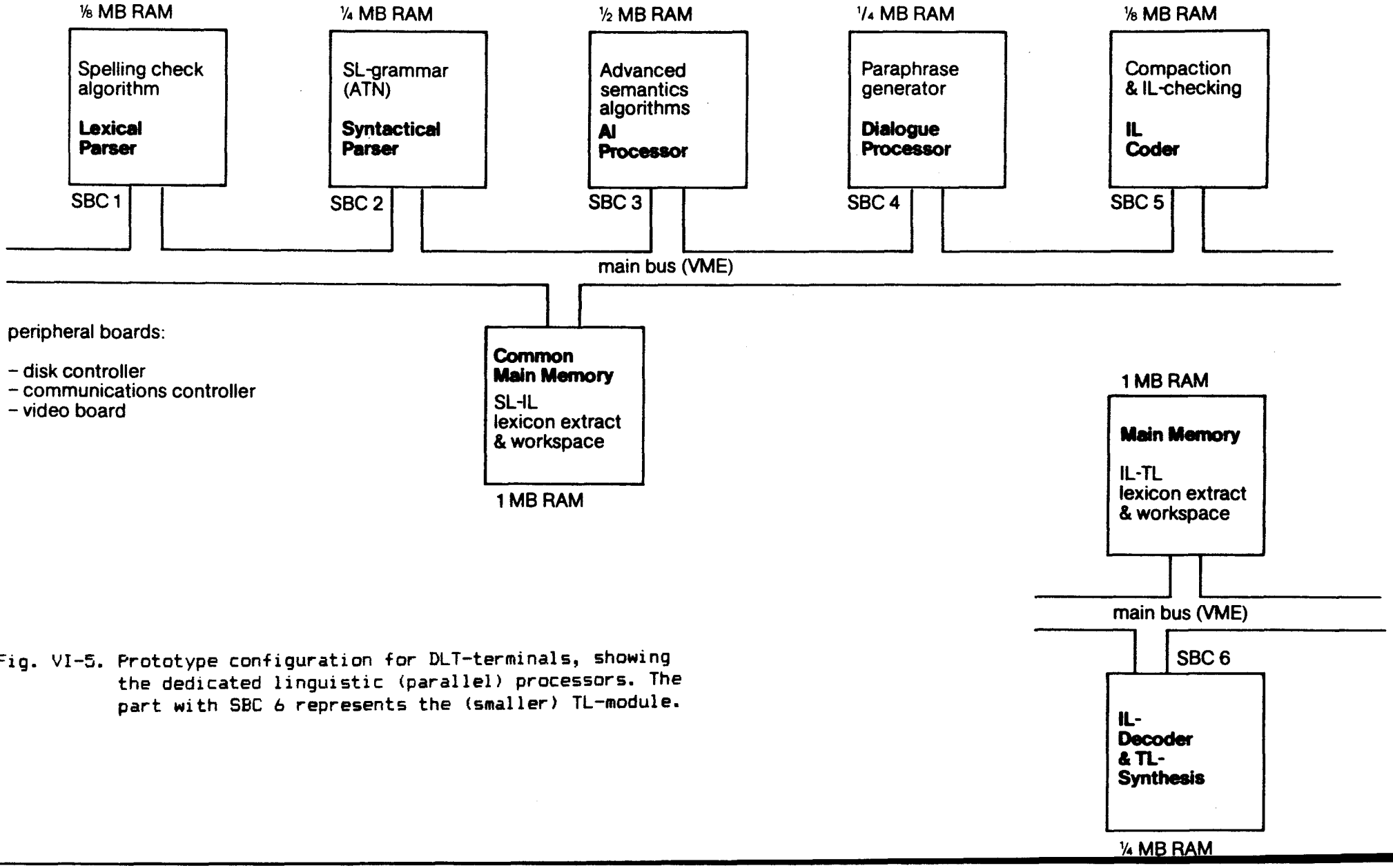


Fig. VI-5. Prototype configuration for DLT-terminals, showing the dedicated linguistic (parallel) processors. The part with SBC 6 represents the (smaller) TL-module.

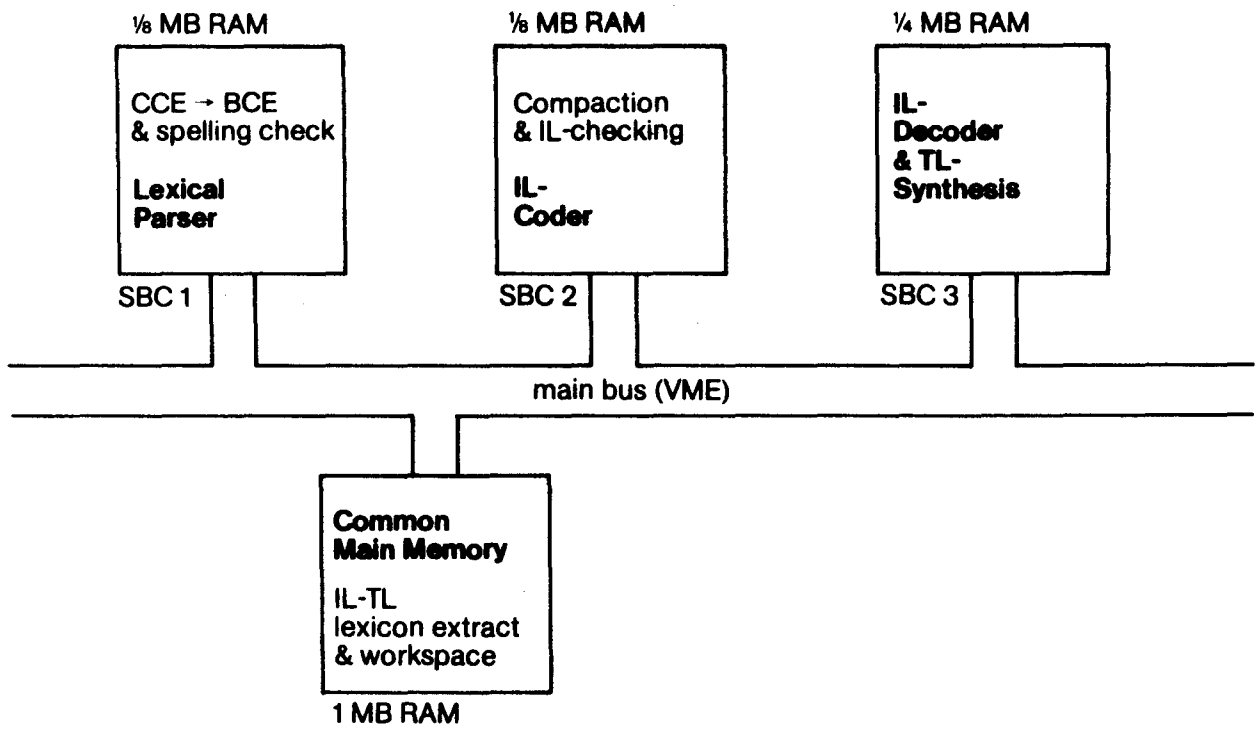


Fig. VI-6a. Initial use of the prototype hardware for the development and testing of the IL-kernel and the IL-to-TL translation stretch.

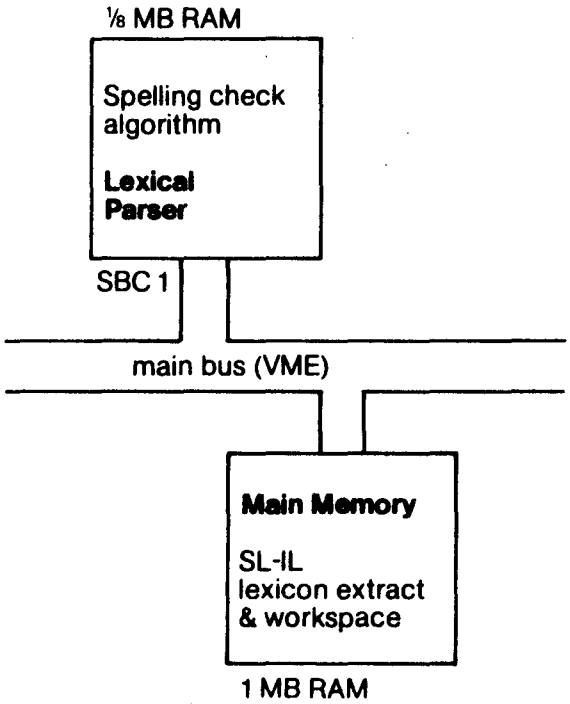


Fig. VI-6b.

As part of SL-module development, the lexical parser with its spelling check program can be tested separately on the prototype.

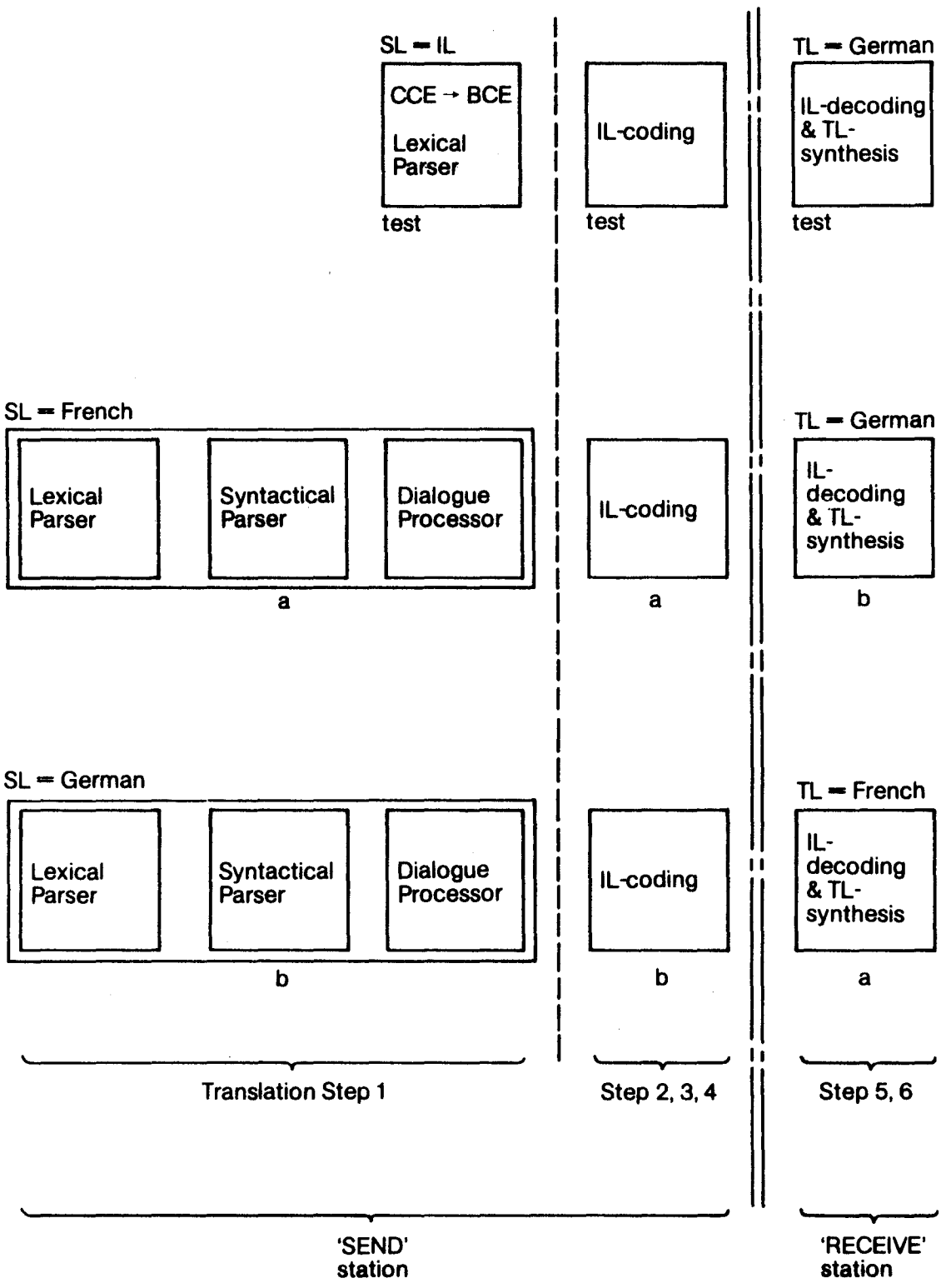


Fig. VI-7. Modularity of DLT development. Each square corresponds with an SBC in the prototype, and each horizontal row represents a translation process. The double chain-dashed line indicates the transmission interface between (remote) terminals, a and b. The single dashed line shows the development interface for different SL-teams.

(extended) IL-kernel [fig. VI-6a]. Useful DLT-functions like lexical parsing with spelling check can be tried out separately [fig. VI-6b], as the partitioning of the prototype hardware well reflects the modularity of the DLT process [fig. VI-7].

One of the objects of the prototype is to investigate several cost-performance trade-offs and to establish parameters such as processing capacity, memory capacity, bus traffic etc. more accurately, in order to make a cost-effective design for the final DLT product, aimed at quantity production. The final product will no doubt be more compact and economic, and its hardware will be more reliable (due to the decreased number of interconnections).

The prospects for a low-priced desktop DLT terminal in the next decade are reinforced by the current technology outlook. Memory density on chips is doubling every two years and is expected to reach 4 Mbit (0.5 MB) by 1990 [Bursky, 1983: 87]. Existing processors will become faster, more versatile, smaller (more highly integrated) and cheaper. Increasing compactness and decreasing price also appears to be a continuing trend in Winchester technology, and may be expected for optical disk units as well.
Based on an assumed reduction of all DLT processing and semi-conductor memories (occupying 6 SBC's in the prototype) to just one 'DLT-board' in 1990, the hardware costs for a DLT-terminal may well come within the Dfl. 20.000-40.000 range (projected retail price, not counting inflation).

4. Software development.

4.1. The choice of PROLOG.

As a vehicle for the development and maintenance of the large amount of DLT-software (450-550 KB, not counting the lexicons and the AI-extension), PROLOG has been chosen.

PROLOG is a non-procedural programming or specification language, which originated in Europe in the early 1970's. In the last five years, it increasingly gained ground upon LISP, with which it roughly shares the same application environment: Artificial Intelligence (AI). PROLOG is being used for a diversity of specific applications in industry as well as in academic institutions now [Colmerauer, 1978; Szeredi, 1982; Clark, 1982], in particular in:

- natural language interfaces
- expert systems implementation
- knowledge acquisition systems
- computer-aided design
- modelling and simulation
- formal system specification

A big impact on the future importance of PROLOG has been made by the start of the Japanese Fifth Generation project, in which PROLOG will serve as the basis for a core language to be embedded in all machines [Agapeyeff, 1982; Kowalski, 1982].

PROLOG can be used and has indeed been used for the specification of ATN's [Sciarone, 1983; Wielinga, 1982], and the language is well suited for pattern matching and tree structure operations [McDermott, 1980].
Interpreters are available for PDP-11, VAX and Z80-based hardware [Goodall, 1983] and under development for the Motorola 68000. In addition to interpreters, also compilers exist or are being developed. Implementation languages include C, CDL2 [Szeredi, 1981] and PROLOG itself, the latter usually to implement PROLOG extensions.
Programming environments built around PROLOG (such as MPROLOG [Köves, 1982]) support modular programming and testing, and offer advanced file handling and editing facilities.
A comparison between PROLOG and LISP [Warren, 1977] revealed that, without a significant loss in efficiency, PROLOG programs are generally more compact and transparent.

As a tool for DLT development, PROLOG will provide an interface to the participating linguists, who will use it as their standard method to specify grammars, tree structures, micro-contextual patterns etc., without being distracted by computer-related programming details and peculiarities. Similarly, PROLOG can be used to specify the structure of dictionary entries, valency subentries etc. [so far as this does not collide with the take-over of the dictionary-building tool COMSKEE from Saarbrücken, as suggested in Chapter VII, W.P. IIa].
Thus, in accordance with the nowadays widely adhered principle of separation between linguistic data and algorithmic processes [Hutchins, 1982], the choice of PROLOG aims at long-term ease of updating and maintenance.

4.2. Practical considerations.

To develop, produce and distribute DLT 'systems', clearly a powerful software development facility will be needed.
After several years of intensive initial development of a first set of SL- and TL-modules, this facility could play the role of a 'DLT software support and distribution center', in addition to the development of further language-modules.
The development machine must be capable of:

* creation, compilation etc. of programs in PROLOG;
* running translation software for test purposes;
* running various linguistic development support software;
* containing the large dictionaries and grammars for all supported languages;
* distibution of new software and dictionaries to DLT users, by means of optical disk or Winchester cartridge;
* distribution of updates by means of data communications (e.g. X25).

For these purposes a machine of the 32-bit 'super-mini' class (DEC VAX, SEL 32, etc.) would be suitable - with a large amount of backing store: currently estimated at basically 100MB, plus another plus 100MB for each language supported.
Also the required number of terminals will depend on the number of languages being developed: 2 terminals for general software development, plus another 2-3 per language (taking

into account lexicon build-up by interactive data entry).

In addition to a Winchester disk of (for instance) 300 MB, a small Winchester or optical disk, similar or equal to the one proposed for the prototype terminal [see sections 1.2 and 3], should also be available on the development machine. Moreover, a line connection directly to the prototype terminal should be provided for down-line software loading and testing. Remote data-communication facilities (incl. X25), a line printer and adequate back-up provisions (presumably a tape drive) should complete the development hardware.

As to software, UNIX is recommended as time-sharing operating system, because of its proven excellence and its increasingly wide acceptance [Tanenbaum, 1982; Evanczuk 1983]. Where needed, C (which normally goes with UNIX) can be used. The choice of PROLOG as a tool for DLT linguistic software development has already been mentioned [see 4.1]. Some of the facilities included in a PROLOG software package (e.g. editing, file handling) may overlap with UNIX facilities, but this should cause no problems.

Finally, a cross-assembly/link facility for the target machine (MC68000) of the terminal prototype will be necessary. The prototype itself should be provided with a real-time multiprocessor operating system, down-line loading and data communications software (X25, Ethernet). It should have its own debugging facilities and a compiler/assembler for low-order systems programming.