сообщения
объединенного
института
ядерных
исследований
дубна

E11-84-656

M.Rudalics, G.Voigt

PL/M BINDING FOR GKS 7.2

1984

0 Purpose and Scope of this Document

This document describes a proposal for binding the functions of the Graphical Kernel Standard (GKS) to the PL/M programming languages PL/M-80 and PL/M-86. According to annex C of the functional description of GKS [1], a binding is defined as the expression of the abstract functions and data types of the standard in terms of constructs available in the host language. Interfacing GKS to the PL/M programming languages may be considered a nontrivial task for two reasons:

- PL/M-80 [2] and PL/M-86 [3] have been specifically designed by one of the main manufactors of microprocessing components for the primary purpose to run on its own hardware, the INTEL 8080 (8008) and 8086 series of microprocessors. PL/M-80 is principally upward compatible to PL/M-86, i.e., a source program written in PL/M-80 may be (with a few restrictions) recompiled with a PL/M-86 compiler. Thus, a common GKS interface may be developed for both languages.

- Due to the poor data type provisions and restricted and unsafe parameter passing facilties of the PL/M languages, a one-to-one mapping of many GKS data and parameter types to PL/M data and parameter types is defeated. This drawback has to be compensated by a cautious use of PL/M's indirect addressing capabilities.

The following chapters are devoted to each of the rules (L1) through (L5) from annex C of the standard and contain the according GKS rule, a mechanism for its realization, and motivations for using the mechanism.

1 GKS Functions - PL/M Procedures Correspondence

Rule (L1): "All GKS functions, other than inquiry functions, shall appear atomic to the application program."

Mechanism: Each GKS function with the exception of some inquiry functions has been realized by one correspondent PL/M procedure. The following substitutions have been made for inquiry functions:

All inquiry functions which return a list or set of values, where the number of elements of the list or the number of members of the set may be altered by the application program, are realized by two procedures, termed NUMBER and VALUE respectively. The parameters of NUMBER correspond to the parameters of the abstract function, only list elements or set members of user defined length or size are omitted. VALUE requires in addition to the parameters of the abstract function an input parameter, which has to be set by the application program to the index of the list element or set member to be investigated. This index has to be in the range (1 .. N), where N is the number of elements of the list or members of the set returned by an invocation of the corresponding NUMBER procedure. VALUE then delivers the value of the element in the list or the member of the set according to this index.

INQUIRE SET OF SEGMENT NAMES IN USE
→ INQUIRE NUMBER OF SEGMENT NAMES IN USE
→ INQUIRE VALUE OF SEGMENT NAMES IN USE

A not predefined pattern representation is inquired in a similar way, i.e., once for each element of the array. The inquire stroke device state function is not substituted. One NUMBER and one VALUE function have been defined for the entire group of functions inquiring the list of output primitives indices. These procedures require one more parameter: the type of the inquired bundle table. One procedure has also been provided for inquiring the default data record of an input device according to the available prompt/echo type. This procedure requires the input class as additional parameter.

Motivations: The scope of an inquiring facility the application programmer has at his disposal, may be defeated by the restriction, that he has to make a wild guess of the length of the list or array (or size of the set) returned by the corresponding function. An obvious consequence would be increased memory requirements, which in their turn may cause considerable problems in microprocessor based implementations. The proposed solution is maybe not quite nice from a software engineering point of view, as it requires the application programmer to calculate the number of times his program has to call VALUE to obtain an entire list, array or set. Therefore it is suggested, that calling VALUE with an invalid index will set the error indicator appropriately.

2 Derivation of PL/M Identifiers

Rule (L2): "The language binding shall specify, for each GKS abstract function name, exactly one identifier acceptable to the language."

Mechanism: For all words constituting abstract GKS function and formal parameter names (inquiry functions have to be modified as indicated in chapter 1) perform the following steps:

(M1) The words POLYLINE and POLYMARKER are substituted by the words LINE and MARKER respectively.

POLYLINE → LINE

(M2) The words LINETYPE and LINEWIDTH are substituted by the strings LINE TYPE and LINE WIDTH respectively.

SET LINEWIDTH → SET LINE WIDTH

(M3) All pronouns, prepostitions, conjunctions and articles, namely ALL, AND, AS, AT, BY, FOR, IN, OF, ON, OR, OUT, THE, THIS, TO, UP, WHICH, WITH, as well as parentheses, commas, and numbers are omitted.

REDRAW ALL SEGMENTS ON WORKSTATION → REDRAW SEGMENTS WORKSTATION

(M4) If after an application of steps (M1) to (M3) still more than six words remain, the trailing words are omitted. (M4) is presently applied to some of the formal parameter names.

MAXIMUM DISPLAY SURFACE SIZE DEVICE COORDINATE UNITS
→ MAXIMUM DISPLAY SURFACE SIZE DEVICE COORDINATE

(M5) All words longer than five characters are truncated on the right to the longest word containing less than or equal five characters which ends with a consonant.

GENERALIZED DRAWING PRIMITIVE → GENER DRAW PRIM

(M6) The remaining words are concatenated. To enhance readability, the concatenation points are indicated by the underline character. Note, that PL/M syntax requires the Dollar sign (which is ignored by the compilers) instead of the underline character.

GENER DRAW PRIM → GENER_DRAW_PRIM

(M7) Procedure identifiers are headed by the letter 'G', parameter identifiers indicating an application controlled memory location by the letter 'P', parameter identifiers denoting a memory location controlled by the implementation are headed by the letter 'Q'.

GENER_DRAW_PRIM → G_GENER_DRAW_PRIM

Motivations: A PL/M identifier may be up to 31 characters in length. For this reason, a simple concatenation of abstract GKS function descriptors would lead to the formation of invalid PL/M procedure identifiers. The wordlength chosen in (M5) raises from the consideration that STRING and STROKE have to be distinguished, while (M7) is explained in more detail in chapter 4. Some unpleasant truncations resulting from an application of (M5), like SWITC, PROMP, RECOR, DEVIC, et al., seem to be more than compensated by the fact, that procedure names may not be confused by corrupting one letter only, or by simply interchanging two letters.

3 Data Types Mapping

Rule (L3): "The language binding shall specify, for each of the GKS data types, a correspondent data type acceptable to the host language, except where convenient for the host language, additional data types may be specified in terms of GKS data types."

GKS Data Type	PL/M Data Type	PL/M-80 Literal	PL/M-86 Literal
INTEGER	WORD	'ADDRESS'	
REAL	REAL	'(4) BYTE'	
STRING	WORD	'ADDRESS'	
	CHARS	'(1) BYTE'	'(1) BYTE'
NAME [1])	WORD	'ADDRESS'	
ENUMERATION	BYTE		
DATA RECORD	STRUCTURE		

[1]) File names and connection identifiers are treated as STRINGs.

Fig. 1: GKS → PL/M Data Types Mapping

Mechanism: While PL/M-80 provides only two data types (BYTE and ADDRESS), PL/M-86 has five of them (BYTE, WORD, INTEGER, REAL, and POINTER). The mapping of GKS data types to PL/M data types (including appropriate literal substitutions) is presented in Figure 1.

Motivations: With the exception of the pixel inquiry functions GKS knows nothing like a negative integer. For this reason PL/M-86 integers have been sacrificed to maintain compatibility with PL/M-80; GKS integers are mapped to PL/M words. An implementation may restrict the range of valid values for word variables to the range (1 .. 65534) and use 65535 as the "invalid" value for the pixel inquiry functions (65535 corresponds to -1 in PL/M's unsigned integer arithmetic). The real data type of PL/M-86 is implementation dependent, four bytes for PL/M-80 is the convention used by the 8080 floating point library [4]. Strings are mapped to a word variable indicating the length of the string and to a byte array which contains the character sequence. Names are generally mapped to words, with one exception: File names and connection identifiers are mapped to a word variable indicating the length of the identifier and a byte array containing the identifier as character string. This allows a more flexible adaption to the host operating system. Enumeration data types are mapped to byte values, compound data types are mapped according to the mapping of their constituents, data records are mapped to PL/M structures, ordered pairs are splitted into their components.

4 Parameter Types Mapping

Rule (L4): "The language binding shall specify, for each GKS abstract function, how the corresponding language function is to be invoked, and the means whereby each of the abstract input parameters is transmitted to the language function and each of the abstract output parameters is received from the language function."

Mechanism: PL/M languages provide only one parameter transfer mechanism, namely call-by-value. Thus, an indirect referencing mechanism for returning values to the application program has to be used. This is achieved by a combination of PL/M's pointer facility and based variables, see Figure 2.

Only input parameters of type byte and word are passed directly via the parameter list, all other input parameters are

GKS Input Parameter	PL/M Input Parameter	PL/M Output Parameter
INTEGER	WORD	P_WORD
REAL	P_REAL	P_REAL
STRING	WORD	P_WORD
	P_CHARS	P_CHARS
NAME [1])	WORD	P_WORD
ENUMERATION	BYTE	P_BYTE
DATA RECORD	P_STRUCTURE	Q_STRUCTURE

[1]) File names and connection identifiers are treated as STRINGs.

Figure 2: GKS → PL/M Parameter Types Mapping

passed with the help of the pointer facility - this is indicated by a 'P' before the PL/M type. While the value of an output parameter of simple type - indicated by a 'P' too - is written directly to a location under control of the application program, certain output data types or output data records are passed via an area which is administrated by the implementation: The GKS procedure returns the location of a compound data type or data record with the help of a quoted pointer - indicated by a 'Q' before the type - and the application program may use an array or a structure based at the location referenced by this pointer for retrieving the inquired value. This template mechanism has to be used for data records and strings, lists, arrays, and sets of predefined length or size. The procedure realizing the open GKS function returns an error file identifier, which has to be used by an eventual application dependent error handling procedure.

Motivations: The concept of quoted pointers relieves the application programmer from the problem of allocating a maybe unpredictable amount of space for storing the inquired values of data records, strings, lists, arrays, or sets of implementation or workstation dependent length or size. Lists, arrays and sets of application dependent size are inquired as indicated in chapter 1.

5 Reserved Words

Rule (L5): "The language binding shall specify a set of identifiers, acceptable to the language, which may be used by an implementation for internal communication."

Mechanism: An implementation has to restrict its public identifiers to be used for internal communication to the group of identifiers beginning with the string 'GKS'.

Motivation: The mechanism follows the recommendation of the standard.

Editorial Remarks: The scope of this publication did not permit a reproduction of the complete PL/M binding. Reference [5] contains the declarations for all PL/M procedures realizing the functions of the standard as used by an application program with the help of the compiler's include directive. Additionally, [5] contains a list of all literal declarations used for the mapping of enumeration data types. All procedure declarations as well as the literal declarations have been checked by the PL/M-80 and PL/M-86 compilers. In a final editing step all Dollar signs have been substituted by the underline character as explained in chapter 2.

References:

[1] Draft International Standard ISO/DIS 7942, Information Processing Graphical Kernel System (GKS), Functional Description, Version 7.2, NI-5.9/1-83, Nov. 1982.

[2] PL/M-80 Programming Manual, INTEL Doc. 98-268, 1976.

[3] PL/M-86 Programming Manual, INTEL Doc. 98-466, 1978.

[4] 8080/8085 Floating Point Arithmetic Library User's Manual, INTEL Doc. 98-452, 1977.

[5] Rudalics, M., and Voigt, G. PL/M Binding for GKS 7.2: Reference List. Dubna, 1984.

Received by Publishing Department
on September 27,1984.

Рудалич М., Фогт Г. E11-84-656
PL/M - интерфейс для GKS 7.2

Стандарт Graphical Kernal System (GKS) определяет языково-независимое ядро графической системы. Для интеграции в программный язык, GKS вводится в языковозависимый слой в соответствии с частными правилами этого языка. GKS-интерфейс языка является документом, описывающим как GKS-функции могут быть доступны программам, написанным на специфическом языке.

В данной работе предлагается интерфейс GKS к языкам программирования PL/M-80 и PL/M-86. В функциональном описании GKS имеется пять правил, которые должны быть соблюдены при стыковке GKS с языком программирования. Каждая глава этой работы посвящена одному из этих правил и содержит описание правила, механизм для его реализации и мотивировки выбора этого механизма.

Работа выполнена в Лаборатории вычислительной техники и автоматики ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1984

Rudalics M., Voigt G. E11-84-656
PL/M Binding for GKS 7.2.

The Graphical Kernel System (GKS) defines a language independent nucleus of a graphical system. For integration into a programming language, GKS has to be embedded in a language dependent layer obeying the particular conventions of that language. A GKS language binding is a document, describing how GKS functions may be accessed by programs written in a specific language.

In this paper a binding of GKS to the PL/M programming languages PL/M-80 and PL/M-86 is proposed. The functional description of GKS lists five rules, which have to be observed when binding GKS to a language. Each chapter of this paper is devoted to one of these rules and contains a description of the rule, a mechanism for its realization, and motivations for choosing the mechanism.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1984