Identification

Implementation of PL/I Storage Classes
D. B. Wagner and M. D. McIlroy

Purpose

This Section describes the various places where PL/I variables
and their associated dope and specifiers are stored.
See BP.2 for the form in which they are stored, as well
as a discussion of data, dope, and specifiers. It should
be noted that the data for a varying string is kept in
free storage, and that information about its current location
and length is kept in a fixed location. In the case of
varying strings the storage indicated below is where this
information about the current generation is kept.

Automatic Storage

All variables with automatic storage class are kept in
the stack frame representing the block in which they are
declared. Dope for an automatic variable is in either
the procedure segment or the stack, depending upon whether
it is adjustable or not, and the specifier is always in
the stack.

Automatic variables in embracing blocks are accessed using
the display, a list of pointers to stack levels for all
containing blocks. See BP.3.00 for details of the display.

Internal Static Storage

All internal static variables for an external procedure
are kept in a block of storage located at

    <stat_>|[uqid]

Uqid is a unique identifier generated at compilation time
through a call to the procedure described in BY.14.01.
(This unique identifier is generated by concatenating
the date and time (GMT) in microseconds with the processor
serial number. An identifier generated in this way is
guaranteed to be distinct from any other generated in
the same way on any GE machine anywhere in the world.)

Dope is always in the procedure segment, since static
storage may not be adjustable. Specifiers are always
in <stat_>|[uqid].

The procedure datmk_is used to create the block of storage
in stat_ the first time any internal static quantity is
referred to in the program.   See BP.4.01 for details.

External Static Storage

Dope for external static variables is kept in the procedure
segment, and specifiers are kept in internal static storage.

The storage of the actual data depends upon whether the
variable-name contains a "$".   If beta is the name of
an external static variable, and the name does not contain
a "$", then the data is at

    <stat_>|[beta]

The data for an external static variable whose name is
of the form alpha$beta is at

    <alpha>|[beta]

In either case space for the data is grown upon first
reference using the procedure datmk_.

Controlled Storage, Based and Non-Based: Discussion

See the PL/I manual (IBM form C28-6571-3, pp. 54-55, 75-77,
103-106, 134-135) for a thoroughly vague and unreadable
discussion of controlled storage.   Generations of controlled
variables are explicitly created and destroyed using the
allocate and free statements.   Because of a historical
twiddle the controlled storage class is divided into two
options: based and non-based.

The non-based option of the controlled attribute is declared
using the attribute

    controlled

Only one generation of a non-based controlled variable
may be referred to at any one time. A push-down stack
of generations is kept in static storage. The allocate
statement "pushes" this stack and the free statement "pops"
it. The only generation which may be referred to at any
one time is that at the top of the stack.

To add to the confusion, when a non-based controlled variable
is passed as a parameter the parameter must be declared
in the called procedure with the controlled attribute.
The parameter passed represents the entire stack of generations,
and the called procedure may do more pushing and popping
of the stack.

The based option of the controlled attribute is declared
using the form of the attribute,

    controlled (p)

where p is a pointer-variable. Every allocation of a
based controlled variable sets a specified pointer-variable
to point to the generation of storage allocated. Any
generation of a based variable can be referred to using
the appropriate pointer.

The non-based option of the controlled attribute has no
right to existence. It should have been removed from
the PL/I specifications when the based option was added,
since it is an easily-programmed subcase and not very
useful anyway. It is being included in the Multics PL/I
primarily for the sake of compatibility with other PL/I
systems.

Based Controlled Storage

There may be allocated into areas with any storage class
including controlled. The procedures used for management
of storage in areas are described in BP.4.02.

The pointer to a generation of based storage points to
the data, and the specifier for the variable is created
in the stack whenever it is referred to. Dope is in either
the procedure segment or the stack, depending upon whether
it is adjustable or not.

Non-based Controlled Storage

[We will have to talk to Digitek before specifying the
implementation of non-based storage.]

Summary

The following table summarizes the information given above.

PL/I Storage class	Location of data	Location of specifier	Location of dope
Static	<stat_>	<stat_>	<proc>
Automatic, non-adjustable	<stack>	<stack>	<proc>
Automatic, adjustable	<stack>	<stack>	<stack>
Based, non-adjustable	<free_>	<stack>	<proc >
Based (area), non-adjustable	area	<stack>	<proc >
Based, adjustable	<free_>	<stack>	<stack>
Based (area), adjustable	area	<stack>	<stack>