Custom Compiler

Develop a compiler for the following language using lex/yacc or any other equivalent.

The following is the description of the constructs supported in the language. At the top of the hierarchy is a *system*. A system contains a process. We now define what a process is. A process has a set of local declaration in a code block and has a start node and an end node which are markers of the process. It supports the following constructs:

- BLOCK - A block of code
- DECISION Construct
- GOTO construct

The block of code is a piece of C-code which may or may not use the variables declared locally in the process. Every block is marked by an identifier which uniquely identifies the code block in the process space. A decision construct decides on the control flow of the language. The GOTO construct is an unconditional jump construct. All jumps are within the process space.

We now define the syntax of the specification: (All keywords are in CAPS)

SYSTEM sysid;
//sysid - System Identifier

SIGNAL sigid1, sigid2, sigid3;
//sigid - Signal Identifier

PROCESS pid1, pid2, pid3;
//pid - Process Identifier

PROCESSDEF pid1 //
Process Definition
START
PROCESSBODY
END

PROCESSBODY as defined earlier is a collection of the four kinds of constructs. We define these constructs in more detail now:

<u>BLOCK construct:</u>
BLOCK blockid //Block identifier
START
Standard C code - Anything
END

If the blockidentifier blockid is not defined in the process scope, it is an error condition. C code is standard C-code.

DECISION construct:

DECISION decisionid //Decision identifier
IF condition
THEN
GOTO blockid_true

Optional - ELSE

Optional - GOTO blockid_false

If the blockidentifiers blockid_false and blockid_true are not defined in the process scope, it is an error condition.

UNCONDITIONAL JUMP:

This is a jump with just one GOTO statement. The syntax is:
UNCONDDEC undecid //Identifier to this block
GOTO blockidentifier

If the blockidentifier is not defined in the process scope, it is an error condition.