INIT$ - Initialise Control Block

The INIT$ routine initialises a Control Block, setting computational and date fields to binary zero; display numeric fields to SPACES, pointer fields to HIGH-VALUES and character fields to a specified Initialisation Character.

1. Invocation

To initalise a control block code:

```
      CALL INIT$ USING cb CB [ic]
```

where *cb* is the control block to be initialised; *CB* is a PIC X(6) field (or 6 character literal padded with SPACESs if necessary) containing the **name** of the control block to be initialised; and *ic* is an optional Initialisation Character.

2. STOP Codes and Exception Conditions

The following STOP codes are generated by INIT$:

STOP code	**Description**
12504	INIT$ was unable to locate the control block, or a symbol within the control block, in the 32-bit Symbol Table.
12505	A zero-length Group Item has been detected in the 32-bit Symbol Table. This should not occur - suspect program corruption.

No exception conditions are returned by INIT$.

3. Programming Notes

INIT$ commences by setting the entire control block to the optional Initialisation Character, or SPACES if the Initiation Character is not supplied. This ensures that all character fields and FILLER's are initialised.

The in-memory copy of the symbol table is then analysed and the following types of field are initialised as follows:

PIC 9(x) COMP	Binary zeroes
PIC 9(x,y) COMP	Binary zeroes
PIC S9(x) COMP	Binary zeroes
PIC S9(x,y) COMP	Binary zeroes
PIC 9(x)	SPACES
PIC 9(x,y)	SPACES
PIC S9(x)	SPACES
PIC S9(x,y)	SPACES
PIC DATE	Binary zeroes
PIC PTR	HIGH-VALUES

Repeating groups, and fields within repeating groups, are initialised correctly.

INIT$ analyses the in-memory copy of the 32-bit Symbol Table to determine which offsets within the control block are to be initialised. INIT$ can take a considerable time to complete if the symbol table is large. If a control block is initialised repeatedly, within a program loop, you are advised to use INIT$ outside the loop to initialise a "template" copy of the required control block; and move the entire "template" control block to the "live" block repeatedly, inside the loop.

The results will be **dramatically unpredictable** if the name of the control block is different from the control block passed to INIT$.

4. Examples

To initialise the AB control block, setting all character fields and FILLER's to SPACES, code:

```
        CALL INIT$ USING AB "AB    "
```

To initialise the ABCDEF control block, setting all character fields and FILLER's to ASCII "Z", code:

```
        CALL INIT$ USING ABCDEF "ABCEF" "Z"
```

5. Copy-Books

None.

6. See Also

B$INIR	Initialise non-DBX database record
B$DXIN	Initialise DBX database record