ISS 45

ICL

CustomLink Technical Reference

Version 7.7

ISS45 7.7 CustomLink Technical Reference

Date of Issue	Product Identification Number	Part Number	Brief Description
August 1995	45001/014	80316779	Version 7.1
February 1999	45001/014	Electronic Library 80602986	Version 7.6
August 2000	45001/014	89000060	Version 7.7 (Unchanged)

Address comments and corrections to:

ICL Retail Systems
ISS45 Program Director
2933 Bunker Hill Lane
Suite 101
Santa Clara, CA 95054

This documentation is designed for placement in an ICL binder that can be ordered separately. To order the binder, contact your sales representative. Indicate PIN 45007/002 and/or part number 80192817.

This sheet contains spine cards that can be used to identify the binder for the appropriate documentation. Cut one of the cards along the dotted lines and insert it in the binder's spine pocket. Discard the remaining cards or save them for later use.

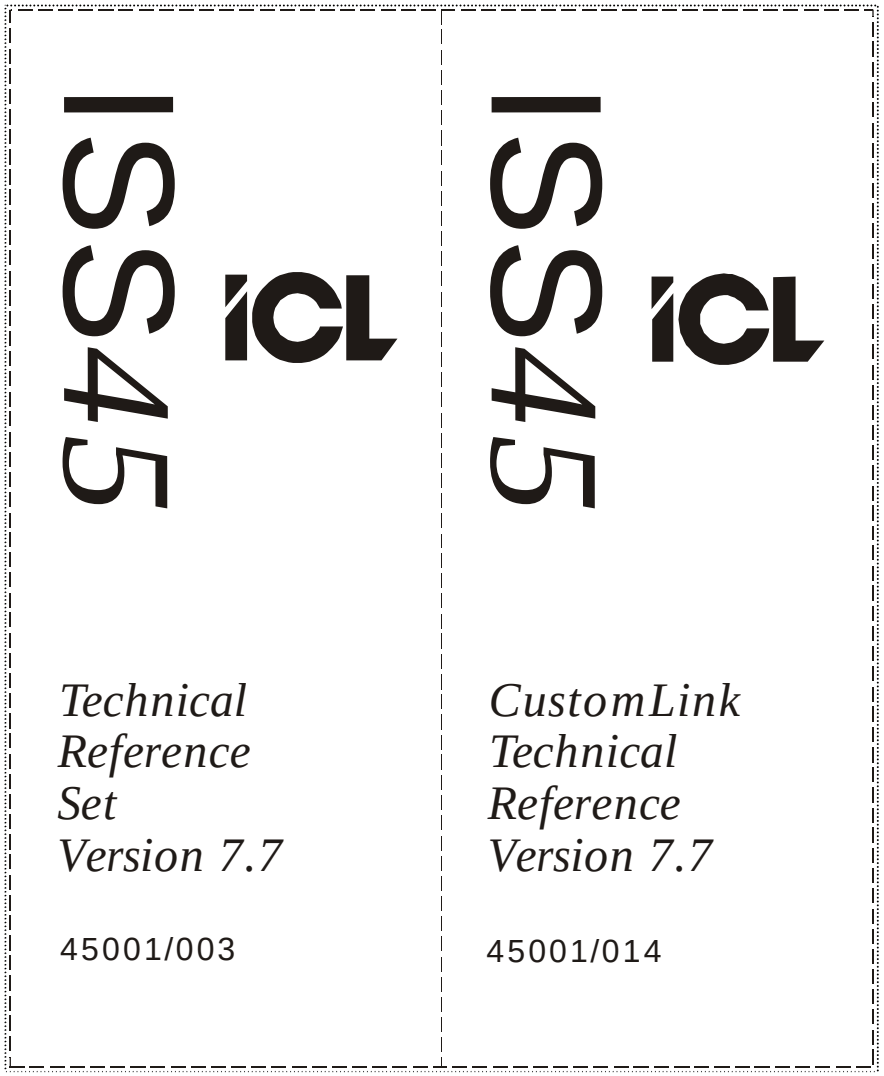

Table of Contents

Introduction

Custom links (hooks), are user-definable applications that can be activated before or after certain functions in the POS terminals.
There are three source files that you can update, according to you own requirements.

File	Description / Use
POSATOM.C	Credit/Check validation. The file includes an example of the SMT interface. (SMOKEY TECHNOLOGY).
POSLANG.C	A language dependent application. Used to convert the amount to text, to be printed on the check face. POSLANG.C is an example of the English version, identical to the source file POSENG.C.
	The file POSSPN.C includes the Spanish version and can be copied over the POSLANG.C to create an application with a Spanish check printout.
POSHOOK.C	Hook functions are self explanatory.
	Please note that you must activate the hooks in the system as well as the specific hook that is used.

To create a **TEAMPOS SELECT** executable file, run: **CMP-SEL.bat**
To create a **TEAMPOS 5000** executable file, run: **CMP-5000.bat**
The result in both cases is the file POS.EXE.

Hooked Functions

hook_init

Description	:	Initialization of the hook application.
Input(s)	:	None.
Output	:	int OK.

hook_before_no_sale_operation

Description	:	This hook is executed just before the start of the next transaction. To be more precise, the hook is executed after the drawer is closed following the previous operation, for example: a sale transaction, pickup, sign-on, etc.
Input(s)	:	None.
Output	:	int OK.

hook_at_end_of_transaction

Description	:	This hook is executed at the end of each sales transaction, but before the printing of the trailer message, and before sending transactions to the host.
Input(s)	:	None.
Globals	:	SALE_TTL structure is defined in 'POSTYPE.H'.
Output	:	int OK.

hook_sign_secure_wait

Description : This hook is executed before sign-on, sign-off, start/stop secure mode and start/stop wait mode.

Value	Meaning
KEY_SIGN_ON	Before sign-on
KEY_SIGN_OFF	Before sign-off
KEY_BREAK	Before or after start of secure mode, depending on the value of the field pp.break_mode.
KEY_WAIT	Before or after start of wait mode, depending on the value of the field pp.wait_mode.

Input(s) : int func.

Output : int OK, or BAD to terminate function execution.

hook_after_total_key

Description	:	This hook is executed immediately after the Subtotal key is pressed, but before any operation is executed.
Input(s)	:	None.
Output	:	int OK.

hook_plu_before_read

Description	:	This hook is executed after entering or scanning the item but before the item information is read from the item file. The return from the hook will determine if PLU sale execution is continued or terminated.
Input(s)	:	struct sale_entry_ *SE - Defined in 'POSTYPE.H'. struct PLU_ *plu- Defined in 'POSSALE.H'.
Output	:	int OK, or BAD to terminate function execution.

Remarks	:	The field code in the PLU structure is packed in BCD format. To unpack, use the following procedure:
		`char plu_code[14];`
		`unpak ( plu_code, plu->code, 14 );`

hook_plu_after_read

Description	:	This hook is executed after the item information is read, but before the information is processed in any way. The return from the hook will determine if the PLU sale execution is continued or terminated.
Input(s)	:	struct sale_entry_ *SE - Defined in 'POSTYPE.H'.
struct PLU_ *plu	-	Defined in 'POSSALE.H'.
int *rc	-	A pointer to the I-O return code, can be OK or !OK.
Output	:	int OK, or BAD to terminate function execution.

hook_plu_after_completion

Description : This hook is executed after the item transaction is competed.

Input(s) : struct sale_entry_ *SE - Defined in 'POSTYPE.H'.

struct PLU_ *plu - Defined in 'POSSALE.H'.

struct BROM_ *brom - Defined in 'POSSALE.H'

(Active promotion structure).

Output : int OK.

hook_dep_before_read

Description	:	This hook is executed after entering a department, but before the department information is read from the item file. The return from the hook will determine if the department sale execution is continued or terminated.
Input(s)	:	struct sale_entry_ *SE - Defined in 'POSTYPE.H'. int *dep_no - Department number.
Output	:	int OK, or BAD to terminate function execution.
Remarks	:	If the department value *dep_no is change, the sale will take affect at the new department.

hook_dep_after_read

Description	:	This hook is executed after the item information is read, but before the information is processed in any way. The return from the hook will determine if the department sale execution is continued or terminated.
Input(s)	:	struct sale_entry_ *SE - Defined in 'POSTYPE.H'. struct DEP_ *dep - Defined in 'POSSALE.H'. int *rc - A pointer to the I-O return code, can be OK or !OK.
Output	:	int OK, or BAD to terminate function execution.
Remarks	:	The department code in the department structure is packed in BCD format. To unpack, use the following procedure:

```
int dep_no;
dep_no = bcd_2_int (dep->no );
```

hook_dep_after_completion

Description	:	This hook is executed after the department sale is competed.
Input(s)	:	struct sale_entry_ *SE - Defined in 'POSTYPE.H'. struct DEP_ *dep - Defined in 'POSSALE.H'.
Output	:	int OK.

hook_disc_before_read

Description	:	This hook is executed after selecting the discount, but before the discount information is read from the discount file. The return from the hook will determine if the discount execution is continued or terminated.
Input(s)	:	int *disc_no-A pointer to the discount number.
Output	:	int OK, or BAD to terminate function execution.
Remarks	:	If the discount value *disc_no is change, the sale will take affect at the new discount.

hook_disc_after_read

Description	:	This hook is executed after the discount information is read, but before the information is processed in any way. The return from the hook will determine if discount execution is continued or terminated.
Input(s)	:	struct DISCOUNT_ *discount - Defined in'POSSALE.H'
		int *rc - A pointer to the I-O return code, can be OK or !OK.
Output	:	int OK.

hook_disc_after_completion

Description	:	This hook is executed after the discount is completed.
Input(s)	:	struct discount_entry_ *DE - Defined in 'POSTYPE.H'. struct DISCOUNT_ *discount - Defined in'POSSALE.H'.
Globals	:	None.
Output	:	int OK.

hook_retn_before_read

Description	:	This hook is executed after selecting the return, but before the return information is read from the returned file. The return from the hook will determine if the return execution is continued or terminated.
Input(s)	:	int *retn_no - A pointer to a return code integer.
Output	:	int OK.
Remarks	:	If the return value *retn_no is change, the sale will take affect at the new return number.

hook_retn_after_read

Description	:	This hook is executed after the return information is read, but before the information is processed in any way. The return from the hook will determine if the return execution is continued or terminated.
Input(s)	:	struct RETURN_ *retn - Defined in'POSSALE.H'. int *rc - A pointer to the I-O return code, can be OK or !OK.
Output	:	int OK, or BAD to terminate function execution.

hook_tndr_before_read

Description	:	This hook is executed after selecting the tender, but before the tender information is read from the tender file. The return from the hook will determine if the tender execution is continued or terminated.
Input(s)	:	int *tender_no-A pointer to the tender number.
Globals	:	None.
Output	:	int OK or BAD to terminate function execution.
Remarks	:	If the tender value *tender_no is change, the sale will take affect at the new tender number.

hook_tndr_after_read

Description	:	This hook is executed after the tender information is read, but before the information is processed in any way. The return from the hook will determine if the tender execution is continued or terminated.
Input(s)	:	struct TENDER_ *tender-Defined in'POSMEDIA.H'. int *rc-A pointer to the I-O return code, can be OK or !OK.
Output	:	int OK.
Remarks	:	The tender code in the tender structure is packed in BCD format. To unpack, use the following procedure:

```
int tender_no;
tender_no = bcd_2_int (tender ->no );
```

hook_tndr_after_completion

Description	:	This hook is executed after tendering is competed.
Input(s)	:	struct tender_entry_ *TE_-Defined in'POSTYPE.H'. struct TENDER_*tender-Defined in'POSMEDIA.H'.
Output	:	int OK.

hook_freq_shopper

Description	:	This hook is executed after the frequent shopper entry, but before it is validated. The return from the hook will determine whether to continue, or re-enter the frequent shopper entry.
Input(s)	:	struct ppm_entry_ *hook_ppm_entry - Defined in 'POSTYPE.H'.
Globals	:	None.
Output	:	int OK, or BAD to terminate function execution.

hook_check_signature

Description	:	This hook is executed just before the 'check signature' prompt. The return from the hook will determine whether to continue or terminate the tender execution.
Input(s)	:	None.
Output	:	int OK, or BAD to terminate function execution.

hook_function

Description : This hook is used for user hook functions.

- The user should put a hook function into the keyboard layout or into the POS menu. Using a hook function will result in execution of this hook with the hook function number.
- The return from the hook will affect the operation of the register.
- If the hook terminates with the OK return code, then the POS will try to execute the function value that exists in the argument 'ppm_entry->code'. It is allowed to put values of POS FUNCTIONS like X_READ or NO_SALE.
- If the hook terminates with the BAD return code, then POS continues normally.

Input(s) : struct ppm_entry_
*hook_ppm_entry - Defined in 'POSTYPE.H'.

Output : int OK, or BAD (see function description).

Remarks : The hook function number is hook_ppm_entry->code - KEY_HOOK_ST + 1;
It is possible to transfer keyed-in numbers into the hook function in the 'long' argument 'ppm_entry->result'.

hook_before_brom_read

Description	:	This hook is executed just before promotion information is read from the BROM file.
Input(s)	:	struct sale_entry_ *SE - defined in 'POSTYPE.H' struct PLU_ *PLU - defined in 'POSSALE.H' struct BROM_ *brom - defined in 'POSSALE.H'
Output	:	int OK or BAD to terminate function execution.
Remarks	:	The return from the hook will determine if the promotion on the item will take place or not. (OK/BAD)

hook_after_brom_read

Description	:	This hook is executed just after promotion information is read from the brom file.
Input(s)	:	struct sale_entry_ *SE - defined in 'POSTYPE.H' struct PLU_ *plu - defined in 'POSSALE.H' struct BROM *brom - defined in 'POSSALE.H' int *rc - a pointer to a return code integer.
Output	:	OK or BAD to terminate function execution.
Remarks	:	The return from the hook will determine if the promotion on the item will take place or not. (OK/BAD). If *rc !=OK, this means that the item was not found.

hook_after_brom_reduction

Description	:	This hook is executed after reduction is completed. Only reductions have an after completion hook, because promotions and offers are executed as a special PLU sale. Therefore the 'PLU after completion' hook covers this purpose.
Input(s)	:	struct sale_entry_ *SE struct PLU_ *plu struct BROM_ *brom
Output	:	int OK.
Remarks	:	

hook_before_maintenance_execution

Description	:	This hook is executed just before every maintenance execution. The return from the hook will determine whether to continue execution of this maintenance (OK/BAD).
Input(s)	:	union maint_ * maint
Output	:	int OK.
Remarks	:	All maintenance types are defined in 'POSMAINT.H'. The general maintenance is defined as follows: Byte opcode: maint->mnt_rec.opcode Datat 96 bytes: maint->mnt_rec.data

hook_after_maintenance_execution

Description	:	This hook is executed just after every maintenance execution.
Input(s)	:	union maint_ *maint
Output	:	int OK.
Remarks	:	All maintenance types are defined in 'POSMAINT.H'. The general maintenance is defined as follows: Byte opcode: maint->mnt_rec.opcode Datat 96 bytes: maint->mnt_rec.data

Functions, Structures & Definitions

Available Function Calls

The following functions are just a few examples of those available. For a full list of functions refer to the 'POSDEC.H' file.

unpak

Description	:	This function unpacks a number from BCD format to ASCII format.	
Input(s)	:	unsigned char *u_str	- Unpacked field.
		unsigned char *p_str	- Packed field.
		int n	- The number of digits in the unpacked field.
Output	:	None.	

pak

Description	:	This function packs a number from ASCII format to BCD format.	
Input(s)	:	unsigned char *u_str	- Unpacked field.
		unsigned char *p_str	- Packed field.
		int n	- The number of digits in the unpacked field.
Output	:	None.	

bcd_2_int

Description	:	This function unpacks a number from BCD format to INT format.
Input(s)	:	char *bcd - Packed field.
Output	:	(int)Value of the BCD field.

Hook_transaction

Description	:	This function writes a user hook transaction.
Input(s)	:	unsigned char sub_opcode. Use only **0x01** that defines the transaction as a user hook. char *data-A 42 character long string that holds the transaction data.
Output	:	None.

system_key

(void);

Description	:	Wait for a keyboard key and background processing continues.
Input(s)	:	None.
Output	:	(unsigned int) value of the pressed key.

Structure Definitions

The following structure definitions and descriptions are only some examples. For a full list of structure definitions refer to the include files

Sale_ttl

Defined in : 'POSTYPE.H'.

struct sale_entry_

Defined in : 'POSTYPE.H'.

SE->ppm_entry.code	Entry source: KEY_ENTER, KEY_SCANNER & KEY_PRESET.
SE->count	Item count.
SE->weight	Item weight (ounces/grams).
SE->decimal_count	Item decimal count (millimeters).
SE->return_item	Next item is returned type 'SE->return_type'.
SE->cancel_item	This is last item cancellation.
SE->subtract_item	Subtract the entered item.
SE->tax_reverse	Sell next item and reverse tax.
SE->FS_reverse	Sell next item and reverse food stamp.
SE->inquery	Item is inquired first, before sold.

struct PLU_

Defined in : 'POSSALE.H'.

struct BROM_

Defined in : 'POSSALE.H'.

struct DEP_

Defined in : 'POSSALE.H'.

struct DISCOUNT_

Defined in : 'POSSALE.H'.

struct discount_entry_

Defined in : 'POSTYPE.H'.

DE->discount_amount	Discount value.
DE->discount_percent	Discount percent.
DE->return_item	The discount is on return item.
DE->cancel_item	The discount is canceled.
DE->subtract_item	The discount is subtracted.
DE->plu_dep_amount	PLU or DEPT. discountable value.
DE->extra_amount	Extra discountable value, affected by any kind of PLU/DEPT. operation, like discount or promotion. The discountable amount is therefore calculated in the following way: Discounttable_value =DE->plu_dep_amount + DE->extra_amount;

struct RETURN_

Defined in : 'POSSALE.H'.

struct TENDER_

Defined in : 'POSMEDIA.H.'

struct tender_entry_

Defined in : 'POSTYPE.H'.

TE->tender_amount	Tender amount (in local currency).
TE->foreign_amount	Foreign currency value.
TE->foreign_rate	Foreign currency rate.
TE->account	Account number.
TE->auth_no	Authorization number.
TE->exp_date_month	Expired month.
TE->exp_date_year	Expiry year.
TE->cancel_tender	Canceled tender.
TE->subtract_tender	Subtracted tender.
TE->change	This is change.
TE->MCR_used	Swipe card was used for this tender.

struct ppm_entry_

Defined in : 'POSTYPE.H'.

Defined values

Defined values are in the include files 'POSDEF.H' & 'POSPPM.H'.

'POSDEF.H' holds the POS application definitions such as:
SALE, TENDER, etc.

'POSPPM.H' holds the general application defininitions such as:
KEY_MENU, KEY_YES, etc.

ICL Retail Systems
2933 Bunker Hill Lane, #101
Santa Clara, CA 95054

P/N 89000060
PIN 45001/014